MASTER IN COMPUTER SCIENCE

Memetic algorithm for discovering biclusters in microarray data

Amant, Stéphane

Award date:
2010

Awarding institution:
University of Namur

Facultés Universitaires Notre-Dame de la Paix Namur
Faculté d'Informatique

Memetic algorithm for discovering biclusters in microarray data

Professeurs : Vanhoof Wim
Divina Federico
Auteur : Amant Stéphane - stephane.amant@gmail.com
Mémoire présenté en vue de l'obtention du grade de Master
en sciences informatiques à finalité spécialisée
Année Académique 2009-2010

Résumé

En biologie cellulaire et moléculaire, la technologie des microarrays (puces à ADN) est un outil fondamental permettant de fournir des informations sur le comportement de milliers de gènes. Dans ce contexte, et plus particulièrement en ce qui concerne l'analyse de leurs données d'expression, un problème important consiste à regrouper les gènes présentant un comportement similaire selon leur activité, c'est-à-dire découvrir des biclusters. Afin de résoudre ce problème, nous nous sommes basés sur une nouvelle approche reposant sur la découverte de biclusters au moyen d'algorithmes génétiques en combinaison avec les idées de Cheng & Church (mean squared residue, row variance, etc). Plus particulièrement, notre travail s'est basé sur l'un des algorithmes de cette famille : le SMOB [5], un algorithme génétique multi-objectif, développé par les Professeurs Federico Divina et Jesus S. Aguilar-Ruiz. Notre approche a donc consisté à améliorer cet algorithme en y incorporant des procédures de recherche locale basées sur les algorithmes de Cheng & Church ainsi que des opérateurs génétiques intelligents. Ensuite, afin de déterminer la qualité de cette nouvelle méthode, des tests ont été réalisés en utilisant trois jeux de données nommés Yeast, Human Lymphoma et Colon Cancer dataset. Finalement, au vu des résultats obtenus, on peut donc conclure que notre approche fonctionne correctement et répond parfaitement aux attentes en améliorant globalement le SMOB classique.

Mots-clés : Microarray, Données d'expression de gènes, Biclustering, Algorithme génétique multi-objectif

Abstract

In cell and molecular biology, the microarrays technology (DNA chips) is a fundamental tool to provide information about the behavior of thousands of genes. In this context, and particularly as regards to the analysis of their expression data, a major problem is to group genes with similar behavior according to their activity, namely "discover biclusters". To resolve this problem, we relied on a new approach based on the discovery of biclusters using genetic algorithms in combination with the ideas of Cheng & Church (mean squared residue, row variance, etc). Our work is based on an algorithm of this family : the SMOB [5], a multi-objective genetic algorithm, developed by Professors Federico Divina and Jesus S. Aguilar-Ruiz. The purpose of our approach is to improve this algorithm by incorporating local search procedures based on the formula of Cheng & Church and intelligent genetic operators. Then, in order to determine the quality of this new method, several tests were conducted using three datasets named Yeast, Human Lymphoma and the Colon Cancer dataset. Finally, from the obtained results, we can conclude that our approach works properly and fully meets expectations by improving the overall classic SMOB.

Keywords : Microarray, Gene expression data, Biclustering, Multi-objective evolutionary algorithm

Remerciements

Au terme d'un travail de longue haleine, il est de tradition de remercier ceux qui, directement ou indirectement, y ont participé.

Ce mémoire est l'aboutissement d'une aventure qui a commencé il y a déjà cinq ans de cela lorsque j'ai franchi la porte de la Faculté d'Informatique.

En premier lieu, je remercie le Professeur Wim Vanhoof d'avoir accepté de me superviser et me soutenir pendant la longue période de préparation de ce travail grâce à ses remarques et conseils avisés, et cela malgré ses nombreuses autres obligations.

Je remercie le Professeur Federico Divina pour la patience qu'il a montrée en essayant de m'initier aux subtilités des algorithmes génétiques, en répondant à mes questions lors de mon stage en Espagne.

Un grand merci aussi à Mmes Céline Dandoy et Isabelle Mostert pour leur relecture attentive que ce soit au niveau du contenu ou de l'orthographe ainsi que pour leurs conseils forts utiles pendant l'élaboration de ce document.

Je tiens également à remercier M. Michael Marcozzi qui grâce à nos discussions m'a particulièrement aidé pendant mon stage et l'implémentation du nouvel algorithme obtenu.

Je n'oublie pas ma famille qui a permis que cette expérience se déroule au mieux.

Enfin, j'ai également une pensée pour tous ceux qui, bien que n'étant pas directement impliqués dans ce travail, m'ont soutenu d'une manière ou d'une autre au cours de cette période.

Table des matières

Chapitre 1

Introduction

Située en biologie et plus particulièrement en biologie cellulaire et moléculaire [76], l'étude des interactions complexes entre les macro-molécules pendant la transcription et les processus de traduction constitue un champ de recherche plein de défis, étant donné leur impact dans de nombreux domaines. Dans ce contexte, la technologie des microarrays (puces à ADN) est un outil fondamental permettant de fournir des informations sur le comportement de milliers de gènes. L'information fournie par cette technologie correspond à l'abondance relative de mRNA (ADN messager) d'un gène donné soumis à une condition expérimentale donnée. Cette information peut être arrangée sous forme de matrice, que l'on appelle "gene expression data matrix" (nommée GEDM ou matrice des données d'expression des gènes), où les lignes et les colonnes correspondent respectivement aux gènes et aux expérimentations [8]. Chaque entrée de la matrice est un nombre réel représentant le niveau d'expression d'un gène donné soumis à une certaine condition expérimentale [7].

Un problème important dans l'analyse des données d'expression des gènes consiste à regrouper les gènes présentant un comportement similaire selon leur niveau d'expression. L'accomplissement de cette tâche aide à mieux comprendre le rôle, la fonction des gènes pendant la transcription des protéines [7] ainsi que dans tous les processus biologiques dont ils font partie. Les microarrays sont donc utilisés dans le domaine médical afin de produire les profils moléculaires des tissus sains et malades des patients. De tel profils sont très utiles pour comprendre au mieux de nombreuses maladies et aident à donner des diagnostics et des traitements plus précis ainsi qu'à découvrir de nouveaux médicaments [3].

La génomique, qui a pour objet l'étude du fonctionnement d'un organisme à l'échelle de son génome [9], peut, grâce à cette technologie des microarrays, révéler les liens entre les gènes. En faisant par exemple l'expérience de désactiver l'un des gènes d'un tissu, les scientifiques observent alors quels sont les gènes réprimés ou quels sont, au contraire, les gènes surexprimés [10]. Ainsi,

des redondances fonctionnelles et des familles de gènes sont décelées. On peut aussi découvrir l'action de nouveaux gènes au sein de mécanismes biologiques mettant en oeuvre des gènes déjà connus [3]. De même, les profils d'expression de différents tissus, de types cellulaires ou même d'un stade de développement cellulaire peuvent être caractérisés via ce processus [10].

On comprend donc à quel point toute amélioration dans ce domaine pourrait apporter une meilleure compréhension du fonctionnement de cellules observées que cela soit de notre corps ou d'un autre organisme. Tout cela ayant ainsi des retombées dans de nombreux domaines comme la médecine et la pharmacie mais aussi dans toutes les sciences ayant de près ou de loin rapport au vivant.

Dans ce contexte, le but de ce travail a consisté plus précisément à élaborer une nouvelle version de l'algorithme génétique multi-objectif SMOB [5] en y intégrant diverses modifications permettant d'obtenir des améliorations significatives par rapport à différentes caractéristiques des résultats obtenus.

Présentons maintenant un aperçu du plan qui a servi de fil conducteur à ce travail.

Nous allons dans un premier temps décrire les différents concepts et procédés utiles à une bonne compréhension du problème et des solutions apportées. Nous les détaillerons plus amplement dans les chapitres suivants. Dans ce cadre, le problème du biclustering et son lien avec la technique des microarrays est abordé en premier lieu. Ensuite, nous citerons les diverses familles d'algorithmes connues permettant de traiter ces problèmes. Puis, nous nous focaliserons plus particulièrement sur une de celles-ci, Greedy iterative search de laquelle notre méthode s'inspire, et présenterons différentes définitions, tirées des articles de Cheng & Church [2], utilisées dans notre approche. Une fois ces étapes préalables accomplies, nous expliquerons alors le fonctionnement des algorithmes génétiques qui constituent l'élément caractéristique de la nouvelle famille d'algorithme traitant les problèmes de biclustering à laquelle appartient le SMOB [5]. Nous présenterons ensuite successivement les algorithmes single objectifs et les algorithmes multi-objectifs qui sont encore plus adaptés au problème à traiter et nous terminerons en décrivant l'étape ultime de leur évolution consistant à incorporer principalement à ces algorithmes génétiques des méthodes de recherche annexes afin d'obtenir une approche hybride encore plus performante.

Ensuite, nous allons présenter les détails de la nouvelle version d'algorithme que nous avons développée et qui a consisté plus précisément à incorporer au sein du SMOB [5] des méthodes de recherche locale basées sur les travaux de Cheng & Church [2] ainsi que des opérateurs génétiques intelligents.

Puis, nous analyserons les résultats obtenus suite à l'application de notre méthode sur trois différents jeux de tests (Yeast, Human Lymphoma et Colon Cancer dataset).

Finalement, nous conclurons ce travail et proposerons diverses pistes pour des améliorations et travaux futurs.

Chapitre 2

Etat de l'art

2.1 Microarrays, Clustering et Biclustering

Cette section a été écrite en se basant principalement sur [11], [4] et [1].

Afin de comprendre au mieux le problème que nous intéresse dans ce travail, commençons par expliquer le fonctionnement des méthodes de microarrays et expliquons ensuite le lien qui existe entre ces procédés biologiques et les méthodes plus mathématiques d'analyse de données qui constituent le coeur de notre problème. Dans ce contexte, nous allons donc détailler le problème du clustering de données d'expression de gènes et plus particulièrement le problème du biclustering de ces données. Nous formaliserons ensuite de manière aussi précise que possible celui-ci, détaillerons les différents types de biclusters et les différentes structures suivant lesquelles ils peuvent s'agencer et expliquerons aussi en quoi ce problème est d'une très grande complexité.

2.1.1 Microarrays

Une puce à ADN, représentée ci-dessous par la figure 2.1, est un ensemble de molécules d'ADN fixées en rangées ordonnées sur une petite surface qui peut être du verre, du silicium ou du plastique. Cette biotechnologie récente permet d'analyser le niveau d'expression des gènes (transcrits) dans une cellule, un tissu, un organe, un organisme ou encore un mélange complexe, à un moment donné et dans un état donné par rapport à un échantillon de référence.

Pour bien comprendre, il faut savoir qu'en biologie, on considère les gènes comme les "plans" des protéines et de certaines autres substances et le mRNA (ADN messager) comme étant le premier intermédiaire lors de la production de n'importe laquelle des molécules génétiquement encodées. C'est pourquoi la concentration en mRNA est généralement appelée le niveau d'expression du gène donné [6].

Après avoir subi l'expérimentation, chaque point (ou "spot") de la puce va être analysé individuellement par un scanner à très haute résolution. L'image scannée va ensuite être traduite de manière informatisée. En fonction de l'intensité du signal il y aura plus ou moins de pixels pour chaque point de la puce.

Voici un exemple : soit un lot de plantes non traitées ainsi qu'un lot de plantes traitées, alors :

- Les gènes dont l'expression est augmentée suite au traitement apparaissent en rouge
- Les gènes peu affectés par le traitement apparaissent en vert
- Les gènes dont l'expression est stable entre les deux conditions apparaissent en jaune (c'est-à-dire rouge + vert)
- Les gènes peu exprimés n'apparaissent pas

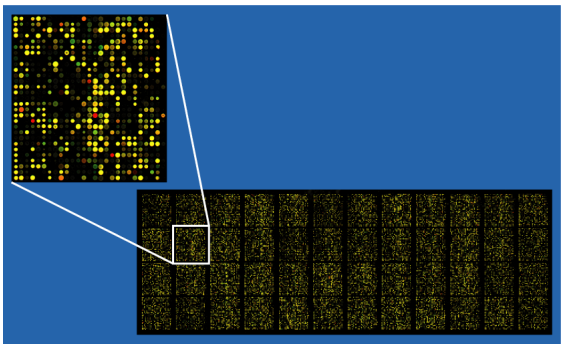


FIGURE 2.1 – Exemple de microarray après expérimentation [11]

On attribue alors à chaque point une valeur d'intensité normalisée par rapport à l'ADN "étalon". Chacune des valeurs peut par la suite être analysée par des techniques de bio-informatique. Ce qui permet d'estimer avec plus ou moins de précision l'intensité d'expression d'un gène.

Les puces à ADN permettent donc de comparer l'expression des gènes de deux types cellulaires différents, d'étudier des gènes exprimés sur un grand nombre de patients pour observer l'effet d'un médicament (anticancéreux par exemple), de regarder l'effet d'un traitement sur l'expression des gènes, de comparer tissus sains contre tissus malades, traités contre non traités, d'étudier l'évolution de l'expression des gènes d'un tissu dans le temps, etc. Cette nouvelle technique constitue ainsi une approche massive et a contribué à la révolution de la génomique.

2.1.2 Clustering et Biclustering

Dans ce contexte, les puces à ADN et autres techniques permettent de mesurer le niveau d'expression d'un grand nombre de gènes soumis à différentes conditions expérimentales. Habituellement, on représente ces données d'expression sous forme d'une matrice, où chaque gène correspond à une ligne et chaque condition expérimentale à une colonne. Chaque élément de la matrice, représentant le niveau d'expression d'un gène soumis à une condition expérimentale, est un nombre réel qui est le logarithme de l'abondance relative de mRNA du gène considéré [1].

Or, visualiser facilement ce type de données et en extraire de nouvelles connaissances biologiques pertinentes est un travail assez difficile [1]. En effet, les objectifs poursuivis par les biologistes lorsqu'ils analysent ces chiffres sont les suivants [1] :

- regrouper les gènes selon leurs expressions sous certaines conditions expérimentales
- classifier un nouveau gène, étant donné son expression et l'expression des autres gènes, via les classifications connues
- regrouper les expérimentations en se basant sur l'expression des différents gènes
- classifier une expérimentation suivant l'expression de gènes soumis à celle-ci

Pour réaliser cela, les techniques de clustering[1] (partitionnement en français) sont les plus appliquées. Elles ont comme objectif principal de regrouper, de partitionner les gènes qui présentent des variations similaires de leur niveau d'expression [5]. Néanmoins, l'application de ces algorithmes de clustering sur les données d'expression fait face à certaines difficultés particulières. En effet, en se basant sur la connaissance globale des processus cellulaires, on comprend vite que certains sous-groupes de gènes ont un comportement similaire seulement sous certaines conditions expérimentales mais qu'ils se comportent indépendamment avec les autres expérimentations [1].

Plus généralement, un cluster est un ensemble de données, similaires les unes aux autres à l'intérieur du même cluster mais différentes aux données des autres clusters [17]. Le clustering a donc pour but de regrouper un ensemble de données en différents paquets homogènes, en ce sens que les données de chaque sous-ensemble partagent des caractéristiques communes [16]. Pour obtenir un bon partitionnement, il convient donc de maximiser les similarités intra-classes pour obtenir des clusters les plus homogènes possibles et minimiser les similarités inter-classes afin d'obtenir des sous-ensembles bien différenciés [3].

1. Ces techniques proviennent en fait des méthodes utilisées dans le cadre du traitement des problèmes de data mining.

Les méthodes de clustering vont être appliquées séparément aux lignes ou aux colonnes des matrices de données. Les méthodes de biclustering, d'un autre côté, réalisent le clustering dans les deux dimensions simultanément. On en tire la constatation que les méthodes de clustering recherchent des modèles globaux alors que les méthodes de biclustering sont ciblées sur des modèles plus locaux. Afin de mieux comprendre, une illustration de ces deux méthodes est donnée ci-dessous par la figure 2.2. Lors de l'utilisation des algorithmes de clustering, chaque gène dans un cluster de gènes donné est défini en utilisant toutes les conditions. De manière similaire, une condition dans un cluster de conditions est définie par les activités de tous les gènes. A l'inverse, chaque gène d'un bicluster est sélectionné en utilisant seulement un sous-ensemble des conditions et chaque condition d'un bicluster est sélectionnée en utilisant seulement un sous-ensemble des gènes. Le but des techniques de biclustering est donc d'identifier des sous-groupes de gènes et des sous-groupes de conditions, en réalisant simultanément le clustering sur les lignes et colonnes de la matrice d'expression des gènes au lieu de réaliser le clustering sur les deux dimensions séparément [1].

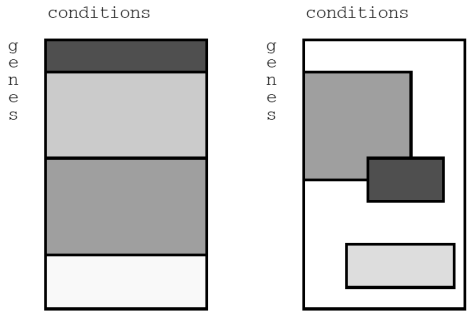


FIGURE 2.2 – A gauche : les méthodes de recherche traditionnelles de clustering qui divisent les gènes en k groupes disjoints. A droite : les méthodes de recherche de biclustering pour un seul ou un ensemble de biclusters contenant chacun un modèle d'expression local [6].

Nous pouvons donc conclure que, au contraire des techniques de clustering, les techniques de biclustering identifient des groupes de gènes qui montrent des modèles d'activités similaires sous un ensemble spécifique de conditions expérimentales. C'est pour cela que l'on est passé de l'application des algorithmes de clustering à celle des algorithmes de biclustering. Etant les plus adaptés, ils sont donc la technique clé à utiliser dans notre situation [1]. Car découvrir ces modèles d'expression particuliers peut mener à de nouvelles découvertes dans le domaine de la génétique. L'utilité de ce concept a d'ailleurs été démontrée dans différentes études ([2] et [18]).

Notons aussi que ce terme de biclustering a été utilisé pour la première fois par Mirkin (1996) [19] pour décrire " le clustering simultané de l'ensemble des lignes et colonnes d'une matrice de données ". Néanmoins, cette idée est bien plus ancienne, on la retrouve pour la première fois dans les travaux de J.A. Hartigan sous le nom de "direct clustering" dés 1972 [20].

2.1.3 Définition et formulation du problème

Un bicluster est défini sur une matrice des données d'expression des gènes, comme représenté par la figure 2.3. Soit G = $\{g_1, ..., g_N\}$ un ensemble de gènes et C = $\{c_1, ..., c_M\}$ un ensemble de conditions expérimentales. Les données peuvent être vues comme la matrice expression de taille N x M appelée EM. EM est une matrice de nombres réels, dans laquelle les valeurs nulles sont possibles et où chaque entrée e_{ij} correspond au logarithme de l'abondance relative en mRNA du gène g_i sous la condition expérimentale c_j. La transformation logarithmique est utilisée afin de convertir les changements multiplicatifs dans la mesure d'abondance en augmentations additives.

Un bicluster correspond alors à un sous-ensemble de lignes qui montrent un comportement similaire par rapport à un sous-ensemble de colonnes et vice-versa [1]. Chaque bicluster peut donc être identifié par un ensemble unique de gènes et de conditions qui déterminent une sous-matrice dans EM montrant une certaine cohérence interne. Un bicluster est une matrice I x J, notée (I, J), où I et J sont respectivement un ensemble de gènes (lignes) et de conditions (colonnes), avec $|I| \leq N$ et $|J| \leq M$. De plus, on définit le volume du bicluster (I, J) comme le nombre d'éléments $e_{ij} \in$ EM tel que $i \in I$ et $j \in J$, c'est-à-dire $|I|$x$|J|$.

	Condition 1	...	Condition j	...	Condition m
Gene 1	e_{11}	...	e_{1j}	...	e_{1m}
Gene ...	...	...	...	...	...
Gene i	e_{i1}	...	e_{ij}	...	e_{im}
Gene ...	...	...	...	...	...
Gene n	e_{n1}	...	e_{nj}	...	e_{nm}

FIGURE 2.3 – Matrice des données d'expression des gènes [1].

Exemple : Supposons que la matrice expression EM soit constituée de 10 gènes et de 8 conditions, comme celle représentée ci-dessous par la figure 2.4. Alors un bicluster défini sur la matrice EM pourrait être ({1, 3, 5}, {2, 4, 7}), c'est-à-dire constitué des gènes g_1, g_3, g_5 et des conditions c_2, c_4, c_7. Le volume de ce bicluster est alors de 9.

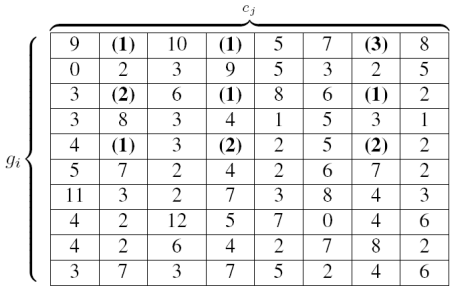


FIGURE 2.4 – Les éléments du bicluster sont en gras entre parenthèses [4].

2.1.4 Types de biclusters

Les biclusters varient suivant le type de cohérence interne que l'on va chercher dans ceux-ci au moyen des algorithmes employés.

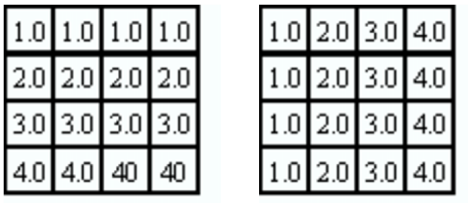
On peut donc identifier les quatre classes principales suivantes :
- les biclusters avec valeur constante
- les biclusters avec valeurs constantes sur les lignes ou les colonnes
- les biclusters avec valeurs cohérentes
- les biclusters avec évolution cohérente

Les biclusters à valeur constante

FIGURE 2.5 – Bicluster à valeur constante [1]

Il s'agit du type de bicluster le plus simple à identifier, tous ses éléments constitutifs sont constants. Il est illustré ici par la figure 2.5. Néanmoins, bien que ce type soit observable lorsqu'on traite des données plus standardisées, dans les données réelles (comme celles venant des microarrays), ils sont généralement masqués par le bruit présent, consécutif du manque de précision des mesures ainsi que des variations présentes dans les résultats obtenus.

Les biclusters à valeurs constantes sur les lignes ou les colonnes



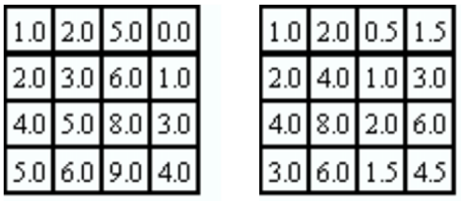
1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0
4.0	4.0	40	40

1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0

FIGURE 2.6 – Bicluster à lignes constantes et à colonnes constantes [1]

De la même manière, ces biclusters présentent ici des valeurs constantes sur leurs lignes ou sur leurs colonnes. On en voit une parfaite illustration via la figure 2.6. Dans le cas de données provenant de l'expression de gènes, un bicluster avec des valeurs constantes sur les lignes identifie un sous-ensemble de gènes ayant des niveaux d'expressions similaires par rapport à un sous-ensemble de conditions expérimentales, si l'on permet que les niveaux d'expression diffèrent pour chacun des gènes. Le même raisonnement peut aussi être appliqué par rapport aux colonnes et aux conditions expérimentales.

Les biclusters à valeurs cohérentes



1.0	2.0	5.0	0.0
2.0	3.0	6.0	1.0
4.0	5.0	8.0	3.0
5.0	6.0	9.0	4.0

1.0	2.0	0.5	1.5
2.0	4.0	1.0	3.0
4.0	8.0	2.0	6.0
3.0	6.0	1.5	4.5

FIGURE 2.7 – Bicluster à valeurs cohérentes (modèle additif et modèle multiplicatif) [1]. Pour le bicluster de gauche, on observe les relations suivantes entre les lignes : $\forall j$, e_{1j} + 1 = e_{2j}, e_{1j} + 3 = e_{3j}, e_{1j} + 4 = e_{4j}. Et pour celui de droite : $\forall j$, e_{1j} x 2 = e_{2j}, e_{1j} x 4 = e_{3j}, e_{1j} x 3 = e_{4j}.

Ce type de bicluster un peu plus sophistiqué se caractérise par le fait que chaque ligne et colonne peut être obtenue en additionnant une constante à chacune des autres ou en multipliant chacune des autres par une valeur constante. Deux exemples sont donnés ci-cessus par la figure 2.7. Via cette approche, on peut trouver des biclusters encore plus complexes dans le cas de données provenant des niveaux d'expression des gènes.

Les biclusters à évolution cohérente

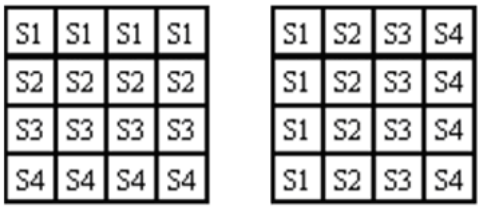


FIGURE 2.8 – Biclusters avec evolution cohérente sur les lignes et sur les colonnes [1]

70	13	19	10
59	40	49	35
40	20	27	15
90	15	20	12

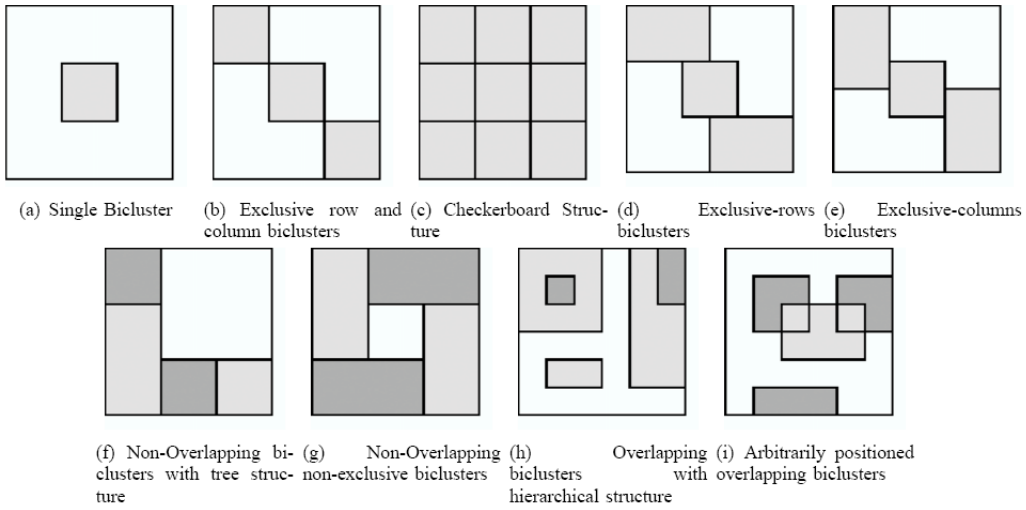
FIGURE 2.9 – Exemple : bicluster avec évolution cohérente sur les colonnes c'est-à-dire que l'on observe la relation suivante entre les différentes colonnes du bicluster : $\forall i,\ e_{i4} \leq e_{i2} \leq e_{i3} \leq e_{i1}$ [1].

Pour ce dernier type de bicluster, illustré par les figures 2.8 et 2.9, on doit voir les éléments de la matrice comme des valeurs symboliques et essayer de découvrir dans cette dernière des sous-ensembles de lignes et des sous-ensembles de colonnes ayant un comportement cohérent sans regard aux valeurs numériques exactes. Dans le cadre de l'analyse des niveaux d'expression, il peut en effet être intéressant d'envisager des cas encore plus particuliers de comportements qu'un sous-ensemble de gènes ait pour un sous-ensemble de conditions sans prendre comme argument principal la valeur actuelle des niveaux d'expression présents dans la matrice.

2.1.5 Structure des biclusters sur la matrice

Parallèlement aux types de biclusters, on distingue aussi différentes sortes de structures que prennent les biclusters dans la matrice d'expression. On a donc adapté les algorithmes utilisés pour le biclustering afin de découvrir ces structures particulières.

Voici les principales structures que l'on retrouve dans les matrices [1] :



(a) Single Bicluster (b) Exclusive row and column biclusters (c) Checkerboard Structure (d) Exclusive-rows biclusters (e) Exclusive-columns biclusters

(f) Non-Overlapping biclusters with tree structure (g) Non-Overlapping non-exclusive biclusters (h) biclusters hierarchical structure Overlapping (i) Arbitrarily positioned with overlapping biclusters

On doit préciser que certaines de ces structures respectent une propriété de visualisation (de (a) à (h)), c'est-à-dire que l'on réordonne les lignes et les colonnes afin de reformer les biclusters visuellement (chacun reçoit une couleur différente et les éléments non classés prennent une couleur neutre). D'autres distinctions peuvent aussi être faites sur les différentes structures : une contrainte d'exclusivité pour l'appartenance d'une ligne ou d'une colonne à un seul bicluster ((b), (d), (e)) ; une contrainte d'exhaustivité obligeant chaque ligne et colonne à appartenir à au moins un bicluster (de (b) à (g)) ; une contrainte de non-chevauchement interdisant qu'un élément appartienne à plus d'un bicluster (de (b) à (g)). On notera néanmoins que toutes ces contraintes se rencontrent difficilement dans des données réelles et que la structure (i) bien plus générale est donc beaucoup mieux adaptée et par conséquent utilisée dans un plus grand nombre d'algorithmes.

2.1.6 Complexité du problème

Bien que la complexité du problème de biclustering dépende de la formulation exacte ainsi que de la fonction utilisée pour évaluer les qualités d'un bicluster donné, presque toutes les variantes intéressantes de ce problème sont NP-complètes. Dans sa forme la plus simple, la matrice de données EM est une matrice binaire où chaque élément e_{ij} est soit 1, soit 0. Si l'on se trouve dans ce cas, un bicluster correspond alors à une biclique (graphe complet biparti [15]) dans le graphe biparti correspondant. Trouver un bicluster de taille maximum est donc équivalent à trouver la plus grande biclique dans un graphe biparti, un problème connu comme étant NP-complet [12]. Les autres cas encore plus complexes, dans lesquels on prend en compte les valeurs numériques de la matrice EM pour calculer la qualité du bicluster, ont donc nécessairement une complexité qui est encore plus élevée. Notons aussi que quelles que soient les caractéristiques des biclusters désirés (pas de chevauchement entre eux, obtenir un ensemble de biclusters au lieu d'un seul, que les biclusters soient construits avec un nombre suffisant de lignes et/ou de colonnes, etc), il a été prouvé que ces problèmes étaient NP-dur ([13], [14]). Etant donné cela, une large majorité des algorithmes utilisent une approche heuristique afin d'identifier les biclusters. Certains d'entre eux, par contre, ne se basent pas sur ces heuristiques mais montrent alors des temps d'exécution exponentiels. En effet, le coût d'une exploration exhaustive de toutes les partitions possibles de l'espace de recherche formant des biclusters est énorme (environ T(N) x T(M)).

2.2 Familles d'algorithmes de biclustering

Cette section a été écrite en se basant principalement sur [1].

Depuis l'introduction du concept de biclustering, de nombreux algorithmes ont été proposés pour résoudre ce problème. On peut classer ceux-ci en différentes familles suivant l'idée fondamentale sur laquelle ils se basent pour aboutir au résultat. Etant donné la complexité du problème, il s'agit bien entendu d'heuristiques. On peut les classer dans les cinq familles suivantes :

- iterative row and column clustering combination
- divide and conquer
- exhaustive bicluster enumeration
- distribution parameter identification
- greedy iterative search

En plus de ces familles, la manière de découvrir les biclusters permet aussi de différencier tous ces algorithmes. Ils peuvent avoir deux objectifs différents : identifier un seul bicluster ou identifier un nombre donné de biclusters. Certaines approches essayent donc d'identifier un bicluster à la fois, d'autres essayent d'identifier un ensemble de biclusters à la fois et d'autres encore essayent d'identifier simultanément tous les biclusters.

Un tableau récapitulatif des algorithmes et de leur classification se trouve en annexe A.

2.2.1 Iterative row and column clustering combination

Il s'agit de la méthode la plus directe pour identifier les biclusters car on réutilise simplement les techniques de clustering standards. On va appliquer séparément les algorithmes de clustering sur les lignes et sur les colonnes de la matrice de données et ensuite combiner les résultats en utilisant certaines procédures itératives sur les deux arrangements de clusters. Un certain nombre d'auteurs ont proposés des méthodes basées sur cette idée, on peut citer entre autres : CTWC [21], ITWC [22] et DCC [23].

2.2.2 Divide and conquer

Cette méthode bien connue en programmation consiste à diviser le problème en plusieurs sous-problèmes similaires à l'original mais de plus petite taille, puis à résoudre ces problèmes récursivement et finalement à recombiner les solutions afin d'obtenir la solution du problème original. Cette approche a l'avantage très intéressant d'être une des plus rapides mais a en même temps le désavantage de rater de bons biclusters car elle les divise avant qu'ils ne soient identifiés. Les

méthodes citées dans la littérature se basant sur cette idée sont l'algorithme de Block Clustering [20] et BiMax [24].

2.2.3 Exhaustive bicluster enumeration

Cette méthode se base sur l'idée que les meilleurs biclusters peuvent seulement être identifiés en utilisant une énumération exhaustive de tous les biclusters existants dans la matrice de données. Ces algorithmes trouvent certainement les meilleurs biclusters s'ils existent mais ont aussi un sérieux désavantage causé par leur complexité. On doit les limiter à ne trouver que des biclusters d'une certaine taille à cause de cela. Différents algorithmes suivent cette voie comme SAMBA [25], pClusters [26] et OP-Clusters [27].

2.2.4 Distribution parameter identification

Cette méthode suppose que les biclusters sont générés en utilisant un modèle statistique donné et essaie donc d'identifier la distribution des paramètres du modèle qui va correspondre le mieux aux données disponibles afin de minimiser, via une approche itérative, un certain critère choisi. On peut citer les algorithmes suivants qui utilisent cette approche : Plaid Models [28], PRMs [29][30] et Gibbs [31].

2.2.5 Greedy iterative search

Un grand nombre d'algorithmes utilisent cette méthode qui consiste à créer un bicluster en ajoutant ou enlevant des lignes ou des colonnes à celui-ci en utilisant un critère qui maximise le gain local de ces opérations. Autrement dit, cette méthode consiste à partir d'un choix local dans l'espoir que celui-ci conduise à une solution globalement bonne. Cette technique peut prendre de mauvaises décisions et perdre ainsi de bons biclusters mais elle a l'avantage d'être très rapide. De nombreux algorithmes sont basés sur cette méthode, on citera les suivants : δ-biclusters [2], FLOC [32][33], δ-patterns [34], OPSMs [36], xMOTIFs [37] et Spectral [35].

2.3 Concepts fondamentaux de Cheng & Church

Cette section a été écrite en se basant principalement sur [4].

Comme nous venons de le voir, parmi les nombreuses méthodes de biclustering proposées dans la littérature, un grand nombre d'approches sont basées sur les heuristiques de type "Greedy iterative search" que nous appellerons plus simplement heuristiques greedy dans la suite. Nous centrerons notre attention sur cette dernière famille d'algorithmes et plus particulièrement sur la méthode et les concepts introduits par Cheng et Church. Cette technique a d'ailleurs été la première à être appliquée sur les données d'expression de gènes [8].

2.3.1 Définitions

Voici les différentes définitions utilisées afin de déterminer une mesure de la qualité d'un bicluster donné via cette approche.

Soit (I, J) un bicluster, alors nous pouvons définir la base d'un gène donné g_i comme $e_{iJ} = \frac{\sum_{j \in J} e_{ij}}{|J|}$. De la même manière, nous pouvons définir la base d'une condition c_j comme $e_{Ij} = \frac{\sum_{i \in I} e_{ij}}{|I|}$. La base d'un bicluster est la moyenne de toutes les entrées contenues dans (I, J), $e_{IJ} = \frac{\sum_{i \in I, j \in J} e_{ij}}{|I|.|J|}$. Signalons que dans les définitions citées ci-dessus e_{iJ} et e_{Ij} correspondent respectivement à la moyenne de la i^e ligne et de la j^e colonne du bicluster (I, J).

Ensuite, parlons du concept de résidu qui quantifie la différence entre la valeur de l'élément e_{ij} et sa valeur attendue comme prédite à partir des moyennes correspondant à sa ligne, sa colonne et celle du bicluster en entier.

Le résidu d'une entrée e_{ij} d'un bicluster (I, J) est $r_{ij} = e_{ij} - e_{iJ} - e_{Ij} + e_{IJ}$.

Le résidu est un indicateur du degré de cohérence d'un élément par rapport aux autres éléments du bicluster étant donné la tendance des gènes et conditions sélectionnés. Plus petit est le résidu, plus forte sera la cohérence.
Notons aussi qu'étant donné cette définition du résidu dans les biclusters et au vu des définitions données précédemment de chacun des types de bicluster, l'utilisation de ces définitions introduites par Cheng & Church au sein de méthodes de recherche va conduire à la découverte de biclusters aux valeurs cohérentes (correspondance entre le modèle additif des biclusters à valeurs cohérentes et la définition de résidu) [1].

Afin d'évaluer la qualité d'un bicluster (au niveau de sa cohérence), le résidu de l'entièreté du bicluster peut être défini comme la moyenne des résidus au carré de tous les éléments spécifiés dans le bicluster, le mean squared residue. Il mesure la cohérence des lignes et des colonnes au sein du bicluster.

Le mean squared residue d'un bicluster (I, J) est $r_{IJ} = \frac{\sum_{i \in I, j \in J} r_{ij}^2}{|I|.|J|}$.

Le mean squared residue donne une indication sur la similarité des données dans la matrice trouvée, si elles sont cohérentes ou aléatoires. Une valeur élevée signifie que les données ne sont pas corrélées et une valeur faible signifie qu'il y a une corrélation dans la matrice. Un score égal à 0 signifie qu'il s'agit d'un bicluster parfait (cohérence parfaite entre les gènes pour l'ensemble des conditions) [38]. Un ensemble de gènes dont les niveaux d'expression changent en s'accordant les uns les autres suivant un ensemble de conditions peut donc former un bicluster parfait même si les valeurs actuelles sont éloignées entre elles [6]. Dès lors, plus bas est le mean squared residue, plus forte est la cohérence du bicluster et meilleure sera la qualité de celui-ci. Si un bicluster a son mean squared residue plus bas qu'une valeur de δ donnée, alors on appellera ce bicluster un δ-bicluster. Ce seuil δ représente la dissimilarité maximale acceptable entre les éléments du bicluster [7].
Il faut également signaler que le mean squared residue défini comme tel suppose qu'il n'y ait pas de valeurs manquantes au sein de la matrice de données. Ce qui n'est malheureusement pas le cas étant donné l'utilisation de microarrays. Donc pour garantir cette pré-condition, on va remplacer les valeurs absentes par des nombres aléatoires durant une phase de calcul préalable [1].

Parlons aussi du cas spécial des biclusters dont le mean squared residue est égal à 0. Ceci est soit dû au fait que le bicluster est constitué d'un seul élément (et dans ce cas on le rejette comme solution possible), ou qu'il s'agit d'un bicluster dont les valeurs fluctuent à l'unisson [1]. Ce dernier type peut donner lieu à des biclusters triviaux ("plats" ou ayant une seule ligne ou une seule colonne) ou alors à de bons biclusters montrant certaines fluctuations particulières. C'est pourquoi on a introduit la mesure supplémentaire de row variance. Car, en général, le but principal du biclustering est de trouver le plus grand bicluster qui n'excède pas une certaine contrainte d'homogénéité. Mais il est aussi important de considérer que la variance de chacune des lignes du bicluster soit relativement haute afin de capturer des gènes montrant des tendances de fluctuations cohérentes sous un ensemble de conditions [7] et donc de rejeter les biclusters triviaux sans signification réelle en biologie (biclusters "plats" cités précédemment) [5].

La row variance d'un bicluster (I, J) est $var_{I,J} = \frac{\sum_{i \in I, j \in J} (e_{ij} - e_{iJ})^2}{|I|.|J|}$.

Dans les analyses de données d'expression de gènes, le but le plus important n'est pas de trouver le plus grand bicluster ou même de trouver la plus grande couverture en biclusters pour la matrice. Le plus important c'est plutôt de trouver un ensemble de gènes montrant des dérégulations similaires (basses ou hautes) par rapport à un ensemble de conditions.

Un faible score de mean squared residue avec une forte valeur de row variance peuvent donc être de bons critères pour identifier ces gènes et conditions. C'est pourquoi, sur base de ces faits, le but recherché par la suite dans ce travail sera toujours de trouver des biclusters de taille maximum, avec un mean squared residue plus bas que le δ donné, avec une row variance relativement haute et avec un faible niveau de chevauchement parmi les biclusters.

Signalons aussi que si nous avions dû aborder d'autres méthodes et d'autres disciplines dans notre travail, les caractéristiques retenues et recherchées permettant de décider de la qualité d'un bicluster auraient pu être totalement différentes.

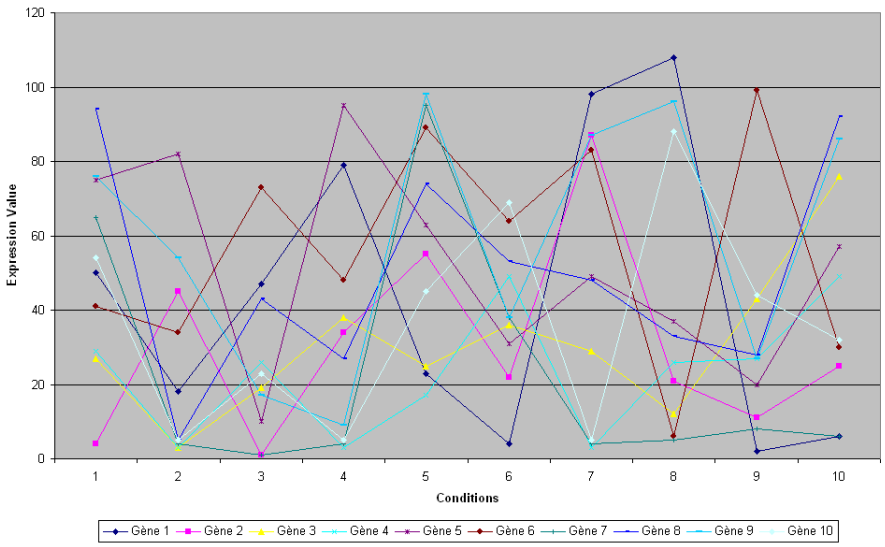
2.3.2 Exemples illustratifs

Voici maintenant quelques exemples à propos de toutes ces définitions afin de donner quelques explications supplémentaires sur les concepts vus précédemment.

Premièrement voici la matrice d'expression des gènes sur laquelle nous allons nous baser. Supposons une matrice expression EM constituée de 10 gènes et de 10 conditions, comme celle représentée ci-dessous, dans laquelle nous avons choisi trois biclusters mis en évidence respectivement en gras, entre parenthèses et souligné.

50	18	47	79	23	4	98	108	2	6
4	45	$\underline{1}$	34	55	$\underline{22}$	87	21	$\underline{11}$	25
27	(3)	19	38	(25)	36	29	(12)	43	76
29	**3**	26	**3**	17	49	**3**	26	27	49
75	82	$\underline{10}$	95	63	$\underline{31}$	49	37	$\underline{20}$	57
41	(34)	73	48	(89)	64	83	(6)	99	30
65	**4**	1	**4**	95	38	**4**	5	8	6
94	(5)	43	27	(74)	53	48	(33)	28	92
76	54	$\underline{17}$	9	98	$\underline{38}$	87	96	$\underline{27}$	86
54	**5**	23	**5**	45	69	**5**	88	44	32

Et sous forme graphique :

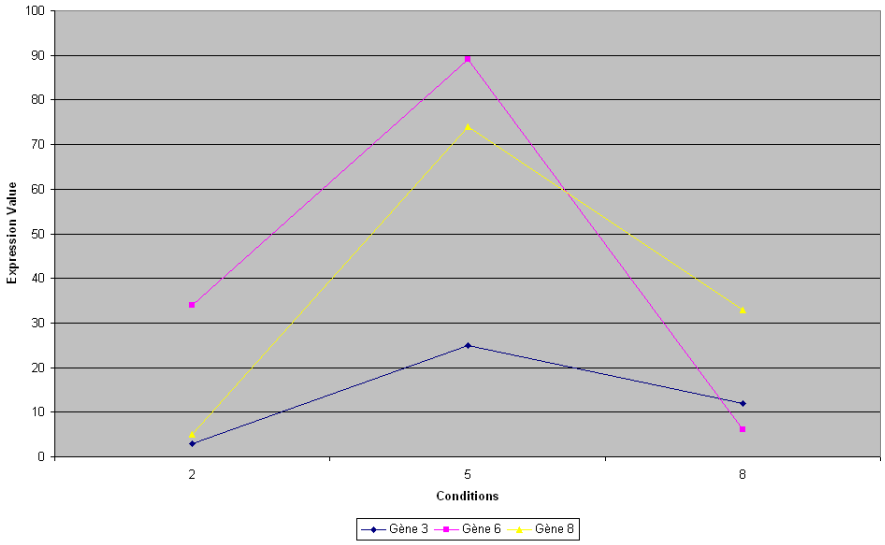


On tire donc de cet ensemble indescriptible et incohérent les trois biclusters suivants.

Soit un bicluster défini sur la matrice EM ({3, 6, 8}, {2, 5, 8}), c'est-à-dire constitué des gènes g_3, g_6, g_8 et des conditions c_2, c_5, c_8. Ce qui donne le bicluster suivant :

$$\begin{matrix} 3 & 25 & 12 \\ 34 & 89 & 6 \\ 5 & 74 & 33 \end{matrix}$$

Et sous forme graphique :



Les moyennes des gènes sont :

- $e_{1J} = 13,33$
- $e_{2J} = 43$
- $e_{3J} = 37,33$

Les moyennes des conditions sont :

- $e_{I1} = 14$
- $e_{I2} = 62,66$
- $e_{I3} = 17$

La moyenne du bicluster est $e_{IJ} = 31,22$.

Voici la matrice des résidus de chacun des éléments :

$$\begin{matrix} 6{,}88 & -19{,}77 & 12{,}88 \\ 8{,}22 & 14{,}55 & -22{,}77 \\ -15{,}11 & 5{,}22 & 9{,}88 \end{matrix}$$

Et finalement voici le mean squared residue $r_{IJ} = 195,16$ et la row variance $var_{I,J} = 691,04$.

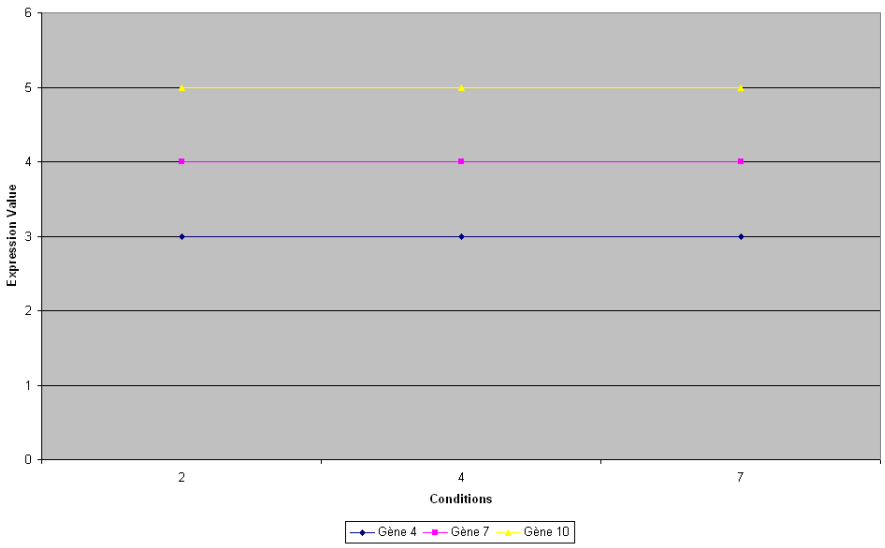
On voit bien que la cohérence entre les gènes n'est pas parfaite mais que, par contre, ces gènes montrent bien des dérégulations sous les différentes conditions. Ce qui explique donc le mean squared residue élevé (pour la mauvaise cohérence) et la row variance élevée (pour les dérégulations observées).

Passons maintenant au cas spécial des biclusters dont le mean squared residue égale 0. Ceux-ci ne sont pas tous à rejeter (excepté les biclusters triviaux d'un seul élément, d'une seule ligne ou d'une seule colonne). En effet, comme nous allons le voir, l'utilisation de la row variance permet de distinguer les biclusters ayant une certaine valeur d'un point de vue biologique de ceux n'en ayant aucune (les biclusters ne montrant aucune variation).

Premièrement, voici un bicluster de cohérence parfaite mais sans signification biologique particulière, car ne montrant aucune variation. Soit un bicluster défini sur la matrice EM ({4, 7, 10}, {2, 4, 7}), c'est-à-dire constitué des gènes g_4, g_7, g_{10} et des conditions c_2, c_4, c_7. Ce qui donne le bicluster suivant (de type bicluster à valeurs constantes sur les lignes) :

$$\begin{matrix} 3 & 3 & 3 \\ 4 & 4 & 4 \\ 5 & 5 & 5 \end{matrix}$$

Et sous forme graphique :

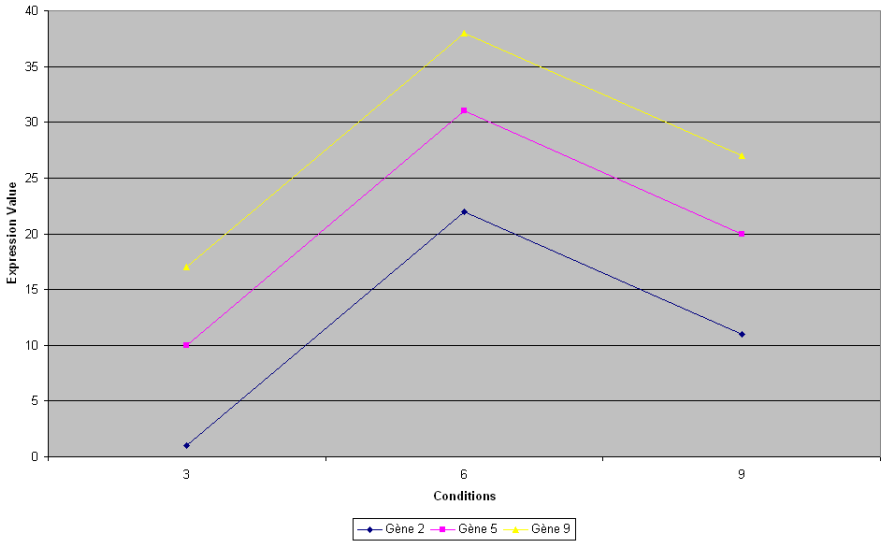


En effet, on observe que le mean squared residue $r_{IJ} = 0$ et que la row variance $var_{I,J} = 0$.

Ensuite, voici un bicluster de cohérence parfaite mais qui a, par contre, une certaine valeur scientifique car il montre des variations. Soit un bicluster défini sur la matrice EM ({2, 5, 9}, {3, 6, 9}), c'est-à-dire constitué des gènes g_2, g_5, g_9 et des conditions c_3, c_6, c_9. Ce qui donne le bicluster suivant (de type bicluster à valeurs cohérentes) :

$$\begin{matrix} 1 & 22 & 11 \\ 10 & 31 & 20 \\ 17 & 38 & 27 \end{matrix}$$

Et sous forme graphique :



En effet, on observe que le mean squared residue $r_{IJ} = 0$ et que la row variance $var_{I,J} = 73,55$.

2.4 Exploration par algorithme génétique

Cette section a été écrite en se basant principalement sur [4], [40] et [41].

Dernièrement une nouvelle approche a été utilisée pour résoudre le problème du biclustering, l'utilisation d'algorithmes génétiques. En effet, il a été prouvé que cette méthode avait d'excellentes performances sur les problèmes d'optimisation fortement complexes à grands espaces de solutions. Deux caractéristiques des algorithmes génétiques ont motivé plus particulièrement leur utilisation : leur excellent pouvoir d'exploration qui leur donne la possibilité de s'échapper des minima locaux et de parcourir bien en profondeur l'espace de recherche ainsi que leur habileté à travailler convenablement même quand les solutions du problème sont soumises à des éléments aux interactions complexes, c'est-à-dire où l'impact de chaque partie sur la solution globale est difficile à modéliser. Ce que l'on retrouve en effet dans le problème du biclustering où l'on doit tenir compte de chaque gènes et conditions et de leurs interactions afin de déterminer la qualité du bicluster. De plus, les algorithmes génétiques, inspirés par les théories de l'évolution, sont connus pour être une méthode de recherche robuste et efficace à adapter aux systèmes biologiques. C'est pour cette raison qu'ils représentent une alternative valide aux heuristiques greedy. Notons que la première tentative de résolution du problème du biclustering de données provenant de microarrays au moyen d'algorithmes génétiques a été proposée par [6].

2.4.1 Les algorithmes génétiques (GAs)

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes (un sous-ensemble des méta-heuristiques). Leur but est d'obtenir une solution approchée, en un temps correct, à un problème d'optimisation, lorsqu'il n'existe pas (ou qu'on ne connaît pas) de méthode exacte pour le résoudre en un temps raisonnable. Les algorithmes génétiques utilisent la notion de sélection naturelle (pression de la sélection et survie du plus apte) développée au XIXe siècle par le scientifique Darwin (dans son livre "l'Origine des espèces") et l'appliquent à une population de solutions potentielles au problème donné [39].

Avant de commencer à détailler plus particulièrement le fonctionnement des algorithmes génétiques, il est important de préciser qu'il ne s'agit ici que d'une présentation partielle du sujet. En effet, suivant les désirs et besoins du concepteur de l'algorithme et du problème à traiter, les différents éléments constitutifs de ce procédé peuvent être combinés de différentes manières et suivant des versions particulières. Il n'est donc pas rare que pour cette même famille d'algorithmes, de multiples implémentations puissent exister, tant la liberté et les possibilités sont grandes. On signalera qu'il existe quand même des versions standards de ces algorithmes ayant chacune leurs avantages et inconvénients, on citera : NSGA-II [45], SPEA2 [44], IBEA [46], NPGA [43] et VEGA [42].

Les algorithmes génétiques résolvent les problèmes difficiles en faisant évoluer les solutions approximatives du problème d'optimisation jusqu'à atteindre une solution. Etant donné un problème d'optimisation, les algorithmes génétiques démarrent généralement d'un ensemble, appelé population, de solutions candidates (souvent choisies au hasard). On fait ensuite évoluer ces solutions au moyen de sélections et de variations sur les solutions les meilleures, en suivant le principe de la survie du plus apte. On se rapproche par "bonds" successifs d'une solution au problème.

Les éléments de la population, qui représentent les solutions candidates, sont appelés individus ou chromosomes. Ces solutions peuvent être encodées de différentes manières. La manière la plus classique consiste à les encoder au moyen de chaînes de bits où chaque bit et sa position dans la chaîne ont une signification particulière. Mais il est aussi possible, en respectant le même principe, d'utiliser des entiers, des réels, des caractères ou tout autre code significatif pour l'application. Cet encodage s'appelle le génotype, au contraire du phénotype qui représente l'objet formé par la solution correspondante dans le contexte du problème original. A chaque génotype correspond au moins un phénotype (la plupart du temps, un et un seul), de telle manière que l'encodage choisi puisse être inversé, c'est-à-dire que le génotype puisse être décodé.

Les individus sont généralement choisis suivant la qualité de la solution qu'ils représentent et afin de réaliser cette mesure, une fonction de fitness est déterminée permettant l'évaluation de chaque individu de la population. De cette manière, meilleure est la fitness d'un individu, plus grande est la possibilité que cet individu soit sélectionné pour la reproduction et donc qu'une grande partie de son matériel génétique soit transmis aux générations suivantes. Le problème pour l'algorithme génétique est donc de détecter les groupes de gènes qui contribuent le plus fortement à la fitness et de trouver leurs bonnes valeurs. Ainsi, au fil des générations, la valeur de fitness des individus va s'améliorer.

Précisons que la fonction de fitness peut prendre plusieurs formes et que suivant la nature de cette fonction, on parlera d'algorithmes génétiques "single objectifs" qui ne donnent qu'une seule réponse au final, au contraire des algorithmes génétiques "multi-objectifs" qui produisent un ensemble de solutions simultanément. Une fonction de fitness "single objectif" agrège en une fonction unique les différentes caractéristiques que l'on veut optimiser par notre calcul. Au contraire, une fonction de fitness "multi-objectif" tente d'optimiser simultanément ces caractéristiques afin de trouver le meilleur compromis[2].

2. Nous aborderons ce type d'algorithme "multi-objectif" plus amplement par la suite.

L'évolution est souvent vue comme le processus d'adaptation à un environnement. De même, la fitness peut aussi être vue comme une mesure de cette adaptation aux exigences de l'environnement. Plus la valeur de la fitness d'un individu est bonne, c'est-à-dire haute ou basse selon la définition de la fonction et l'objectif recherché, plus cet individu rencontre les exigences demandées. Il aura par conséquent plus de chances de se reproduire. Ainsi, au fur et à mesure des générations, la population va être de plus en plus adaptée à son environnement. Cela signifie donc, dans notre contexte d'algorithmes génétiques, que la population va se rapprocher de plus en plus de la solution.

Un autre point important pour les algorithmes génétiques est de maintenir la diversité de la population [57]. Maintenir cette diversité permet d'avoir une répartition des individus sur tout l'espace de recherche afin qu'il soit exploré dans son entièreté et qu'aucune région de cet espace ne soit surpeuplée. En effet, il est très important de s'assurer que la population ne converge pas trop rapidement vers un seul individu qui serait un optimum local au détriment d'une recherche plus approfondie [40]. Il vaut mieux maintenir suffisamment longtemps la "diversité génétique" de la population, afin d'éviter une convergence prématurée de celle-ci [56]. La convergence signifie que la valeur moyenne de la fonction de fitness a tendance à se rapprocher de celle de l'individu le plus adapté, signe d'une uniformisation croissante de la population vers cet individu. On va donc faire une recherche plus globale au début, puis à mesure que le temps passe se diriger vers une recherche plus locale sur le ou les individu(s) qui nous intéresse(nt) [39]. Il est donc important de ne pas avoir de régions surpeuplées mais que toutes les régions prometteuses soient explorées [40]. Afin de réaliser cet objectif, on va mettre en place des fonctions calculant des distances entre individus afin d'estimer et de maintenir la diversité de ceux-ci dans la population. Ces différentes méthodes proposées ont toutes des effets sur les mécanismes de sélection des individus et donc sur l'état de la population générée. On va donc incorporer ces mesures de distances de différentes manières pour qu'elles agissent correctement : soit en complément de la fonction de fitness, soit en les liant à l'opérateur de sélection [3].

Les individus sélectionnés se reproduisent au moyen de croisements et de mutations dont on paramètre la fréquence. En termes simples, les croisements échangent le matériel génétique entre deux individus (ou plus) alors que les mutations changent une petite partie du matériel génétique d'un individu. A partir de cette phase de reproduction, de nouveaux individus sont générés. Les progénitures rentrent alors en compétition avec les anciens individus de la population pour une place dans la nouvelle génération. Typiquement, les nouveaux individus vont remplacer certains des pires individus de la population en se basant sur

3. Plus précisement, "en complément de la fonction de fitness" signifie "inclue ou non dans la fonction" dans le cas de fonctions "single objectifs" alors que pour le cas de fonctions "multi-objectifs", cela signifie "en supplément, en parallèle de la fonction".

la fitness. Mais une autre stratégie consiste à remplacer tous les vieux par les nouveaux individus générés.

Le fonctionnement de l'algorithme est donc basé sur l'alternance de deux opérations, les variations (croisements et mutations) ainsi que la sélection naturelle qui va favoriser les individus les plus adaptés et réduire le nombre des mauvais jusqu'à ce qu'un individu de la population (algorithme "single objectif") ou qu'un ensemble d'individus de la population (algorithme "multi-objectif") réalise l'optimum de l'objectif ou que la population stagne sans plus se renouveler. De cette manière, les algorithmes génétiques peuvent efficacement explorer l'espace des solutions possibles du problème d'optimisation. Cet espace est appelé espace de recherche et contient toutes les solutions possibles qui peuvent être encodées. Il est d'ailleurs généralement accepté que les algorithmes génétiques sont très performants pour la recherche dans de grands espaces, ce qui est aussi le cas pour le traitement du problème de biclustering comme nous le verrons plus loin.

Voici le schéma haut niveau d'un algorithme génétique :

```
1   Initialisation de la population
2   Evaluation de chaque individu de la population
3   Repeat
4           Sélection des parents
5           Croisement des parents
6           Mutation des progénitures obtenues
7           Evaluation des progénitures
8           Insertion des progénitures dans la population
9   Until (Critère d'arrêt)
10  Extraction de la solution à partir de la population
```

Dans ce procédé, la première opération faite est l'initialisation de la population. Celle-ci peut être faite au hasard ou suivant différentes stratégies. Ensuite, chaque individu de la population est évalué. On fait ensuite évoluer les individus (à l'intérieur de la boucle). A l'étape 4, un certain nombre d'individus sont sélectionnés dans la population. Ces individus servent ensuite à en générer de nouveaux. Ils sont créés par l'application d'opérateurs de croisement et de mutation aux étapes 5 et 6. Ces deux opérateurs sont appliqués suivant une probabilité donnée. Les nouveaux individus sont ensuite évalués et insérés au sein de la population. Ce processus est itéré pendant un certain nombre de générations jusqu'à ce qu'un critère d'arrêt soit rencontré. Nous détaillerons par la suite les différentes phases citées dans cette brève introduction.

Notons que les algorithmes génétiques sont aussi appliqués à des problèmes se posant dans de nombreux domaines, entre autres : le planning [48], le design [49], les horaires [50], la simulation et l'identification [51], le contrôle [52] et la classification [53].

Signalons enfin que l'origine des algorithmes évolutionnaires remonte aux années 1950, mais il est généralement admis que le fondateur de la théorie des algorithmes génétiques est John Holland, professeur à l'université de Michigan dans les années 1960. Le but initial de Holland ne s'écartait cependant pas de l'étude formelle des populations, notamment animales, soumises à la sélection naturelle et aux opérateurs génétiques (accouplements, mutations, etc). C'est néanmoins un élève de Holland, David Goldberg qui a eu l'idée de faire sortir ces techniques de leur cadre d'origine pour aborder la résolution de problèmes d'algorithmique en copiant le phénomène de l'évolution naturelle pour rechercher la solution à des problèmes mathématiques.

Pour toute question supplémentaire sur le sujet, le lecteur peut se référer à [47].

2.4.2 Détails des composants des GAs

Nous allons présenter dans cette section différentes particularités des algorithmes génétiques que nous avons survolées dans la présentation générale précédente :

- la sélection des individus
- les opérateurs de croisement
- les opérateurs de mutation
- les mesures de distance
- l'élitisme

La sélection des individus

La sélection joue le rôle de filtre de la population : les meilleurs individus sont retenus ou leur nombre est accru, et les plus faibles sont éliminés ou leur nombre est diminué. Pratiquement il faudrait que chaque individu soit retenu ou écarté en fonction de sa fitness [41].

Pour ce faire, il existe de nombreuses façons de procéder mais nous présenterons seulement la méthode du tournoi car elle est une des plus courantes.

Ce mode de sélection va permettre de choisir un individu au sein de la population d'un algorithme génétique via l'utilisation de tournois. C'est-à-dire que la sélection de l'individu va s'effectuer parmi quelques-uns de ceux-ci choisis au hasard dans la population [54]. La méthode du tournoi consiste à sélectionner deux individus ou plus au hasard dans la population afin de les faire participer au tournoi ainsi qu'à choisir un paramètre p, représentant une probabilité d'être sélectionné, compris entre 0,5 et 1. L'individu de fitness la plus forte parmi les individus reçoit la probabilité p d'être choisi au final par le tournoi et les suivants reçoivent des probabilités de plus en plus faibles [41]. Finalement, le choix de l'individu à renvoyer s'effectue en respectant les différentes probabilités fixées pour les individus participants. Néanmoins, un des cas les plus courants est celui où $p = 1$ et dans ce cas, seul le meilleur individu du tournoi est sélectionné. Après la sélection, l'individu choisi peut être enlevé de la population dans laquelle il a été sélectionné mais il peut aussi y rester et être sélectionné à nouveau [54]. La pression de la sélection, c'est-à-dire le degré avec lequel les bons individus sont favorisés, influence l'algorithme génétique dans sa manière d'améliorer la fitness de la population au fil des générations. La vitesse de convergence de l'algorithme génétique est largement déterminée par la pression de la sélection, si celle-ci est trop basse, l'algorithme va converger trop lentement et si celle-ci est trop haute, l'algorithme risque de converger trop vite vers une solution incorrecte qui aurait encore pu être améliorée [55]. Ici, la pression est facilement ajustable dans cette méthode, un changement de la taille du tournoi suffi. Cela signifie que plus la

taille de ce dernier est grande, plus les individus ayant une mauvaise fitness ont une chance d'être sélectionné [54].

Les opérateurs de croisement

Un opérateur de croisement prend deux individus ou plus en entrée et retourne deux enfants ou plus au final. La progéniture est obtenue en échangeant certaines parties provenant des parents. Le choix des parties ainsi que de la manière avec laquelle elles vont être combinées dépend de la méthode utilisée. Le principe derrière le croisement est qu'en unissant deux individus avec des caractéristiques différentes mais désirables, un enfant combinant les bonnes caractéristiques pourrait être créé [40]. De la même manière, certains enfants créés par croisement peuvent aussi être une combinaison des mauvaises parties de leurs parents. Cependant, à cause de leurs défauts, ils ont une faible probabilité d'être sélectionnés pour la reproduction et de transmettre ceux-ci aux générations futures. Via l'utilisation des opérateurs de croisement, la diversité de la population est maintenue, ce qui permet de chercher dans un plus grand nombre de biclusters [4].

Le croisement peut être défini de différentes manières, voici trois exemples fréquemment rencontrés [40] :

- Un exemple du "single point crossover"

Soient deux parents :
$P_1 = 11111|111111111$
$P_2 = 00000|000000000$

Un point est choisi au hasard à l'intérieur des chaînes de bits utilisées pour représenter les deux individus. Il est noté par un |. Après l'application du single point crossover, on obtient les deux enfants suivants :
$E_1 = 11111|000000000$
$E_2 = 00000|111111111$

- Un exemple du "two point crossover"

Soient deux parents :
P_1 = 111|11111111|111
P_2 = 000|00000000|000

Deux points sont choisis au hasard à l'intérieur des chaînes de bits utilisées pour représenter les deux individus. Ils sont notés par un |. Après l'application du two point crossover, on obtient les deux enfants suivants :
E_1 = 111|00000000|111
E_2 = 000|11111111|000

- Un exemple du "uniform crossover"

Soient deux parents :
P_1 = 1|1|1|1|1|1|1|1|1|1|1|1|1|1
P_2 = 0|0|0|0|0|0|0|0|0|0|0|0|0|0

Les points de croisement sont notés par un |. Les échanges se font au hasard. Après l'application du uniform crossover, on pourrait obtenir les deux enfants suivants :
E_1 = 0|1|1|1|0|1|0|1|1|0|0|0|0|1
E_2 = 1|0|0|0|1|0|1|0|0|1|1|1|1|0

Les opérateurs de mutation

Les opérateurs de mutation reçoivent un seul individu en entrée, seulement une petite partie de celui-ci est modifiée et le nouvel individu ainsi obtenu est retourné au final [40]. Cela consiste en fait à changer la valeur d'un des bits choisis au hasard et de lui donner une valeur différente aléatoire elle aussi [41]. Les mutations garantissent que l'on visite au maximum l'espace de recherche, c'est-à-dire que l'on puisse atteindre n'importe lequel des individus durant la recherche [4]. De plus, elles sont les seules opérations à pouvoir introduire un élément inexistant dans la population initiale ou réintroduire ce qui aurait disparu. De la même manière que les opérateurs de croisement, les opérateurs de mutation peuvent créer de bons individus mais aussi de mauvais [41]. Notons que cet opérateur est appliqué sur les nouveaux individus avec une faible probabilité étant donné son côté aléatoire, au contraire de l'opérateur de croisement utilisé beaucoup plus fréquemment dans l'algorithme [4]. Il est donc possible que les enfants obtenus soient égaux aux parents suite à l'application des opérateurs de croisement ou de mutation.

La mutation peut être définie de différentes manières, voici un exemple fréquemment rencontré [40] :

- Un exemple de "classical mutation operator"

Etant donné un individu :
$I = 1111\underline{1}0000001111$

L'opérateur peut changer la valeur d'un bit (celui souligné dans l'exemple) et on obtient ainsi l'individu :
$I = 1111\underline{0}0000001111$

Les mesures de distance

Les calculs de distance servent à mesurer la similarité entre individus. Les fonctions utilisées pour déterminer cette similarité agissent au niveau du génotype ou au niveau du phénotype. Au niveau du génotype, cela signifie que l'on considère seulement la structure de l'encodage des individus pour calculer une distance entre ceux-ci. Si sur l'ensemble de la population, ces distances calculées pour chacun des individus pris deux à deux sont petites et varient peu, cela veut dire que les individus se ressemblent tous et que l'on n'explore qu'une seule région. Cet état de fait peut être bénéfique à la fin du processus de recherche mais être néfaste en débutant celui-ci car il manque alors de diversité dans la population. Au niveau du phénotype, la distance entre deux individus est celle qui existe entre leurs domaines sémantiques, c'est-à-dire la distance entre leurs valeurs objectives, ayant chacune une signification dans le contexte du problème considéré. Cette fois encore, le fait d'avoir de grandes distances indique une bonne diversité dans la population. Néanmoins, ces méthodes permettant d'assurer la diversité nécessitent beaucoup de calculs, car il faut considérer la distance entre chaque paire d'individus de la population.

L'élitisme

Cette stratégie plus récente donne souvent d'excellents résultats. Elle consiste à réserver des places dans la population de la génération suivante pour les meilleurs individus de la population actuelle. Un nombre déterminé, les k meilleurs de la population actuelle, sont retenus et une sélection de r individus s'effectue d'office entre eux avant de compléter la population de la génération suivante $(\forall k, r)$.

Exemple : supposons M = la taille totale de la population ; r = 5 ; k = 3. Les trois meilleurs individus actuels sont retenus pour une sélection de 5 individus dans la population de la génération suivante (par exemple 2 fois le 1^e, 2 fois le second et 1 fois le 3^e) avant de compléter le reste de cette population avec M-5 individus parmi lesquels on peut à nouveau trouver les premiers.

La stratégie élitiste possède différentes variantes qui toutes ont pour but de préserver les meilleures solutions de la population actuelle afin de les passer à la génération suivante. Notons que l'élitisme joue aussi le rôle de maintenir la diversité au sein de la population au fil des générations [8].

2.5 Adaptation des GAs au problème du biclustering

Le problème de trouver un ensemble de biclusters ayant des caractéristiques particulières à l'intérieur d'une matrice de données d'expression peut être vu comme un problème de recherche dans l'espace contenant tous les biclusters qui peuvent être obtenus à partir de la matrice. Néanmoins, ce problème a un espace de recherche énorme, et même pour les algorithmes génétiques, ce n'est pas aussi facile qu'il n'y paraît [4]. Un nouveau type d'approche, développée en premier lieu par [6], consiste en fait à utiliser les algorithmes génétiques en combinaison avec les différentes définitions comme le mean squared residue et la row variance provenant de l'article de Cheng et Church afin d'évaluer les biclusters via la fonction de fitness et explorer ainsi de manière intelligente les solutions possibles du problème. De plus, comme leur fonctionnement se base sur les mêmes définitions, les résultats de cette nouvelle famille d'algorithmes pourront d'ailleurs être comparés avec ceux de Cheng et Church.

Le dernier point restant alors à déterminer est celui de trouver une formulation correcte faisant le lien entre les algorithmes génétiques et le problème du biclustering. On a donc dû choisir un encodage où chaque individu de la population de l'algorithme génétique représente un bicluster, c'est-à-dire une solution potentielle du problème d'optimisation. Les biclusters sont encodés au moyen de chaînes de bits de longueur fixée N + M, où N et M sont respectivement le nombre de lignes (les gènes) et de colonnes (les conditions expérimentales). On construit cette chaîne en mettant à la suite, une chaîne de bits pour les gènes puis une autre pour les conditions. Si un bit de la chaîne est mis à 1, cela signifie que la ligne ou colonne correspondante appartient au bicluster encodé par cette chaîne, sinon il n'est pas pris en compte [4].

Par exemple, supposons la matrice expression EM constituée de 5 gènes et de 5 conditions, comme celle représentée ci-dessous :

$$\begin{matrix} 50 & 18 & 47 & 79 & 23 \\ 4 & 45 & 1 & 34 & 55 \\ 27 & 3 & 19 & 38 & 25 \\ 29 & 3 & 26 & 3 & 17 \\ 75 & 82 & 10 & 95 & 63 \end{matrix}$$

Soit un bicluster défini sur la matrice EM ({2, 4}, {1, 5}), c'est-à-dire constitué des gènes g_2, g_4 et des conditions c_1, c_5. Ce qui donne le bicluster suivant :

$$\begin{matrix} 4 & 55 \\ 29 & 17 \end{matrix}$$

Et sous sa forme encodée, on obtient la chaîne de bits suivante : 01010|10001. Le symbole | est seulement utilisé afin de délimiter les bits relatifs aux lignes de ceux relatifs aux colonnes.

Notons que d'autres choix auraient pu être faits concernant la représentation des individus. Cependant, la plupart des algorithmes de la littérature qui suivent cette approche basée sur les algorithmes génétiques utilisent aussi cet encodage [4, 5, 3, 6]. Signalons donc qu'un autre encodage possible faisant intervenir des lignes inversées dans le calcul des biclusters est aussi utilisé par des algorithmes génétiques traitant le même type de problème [7]. La signification exacte de ces lignes inversées peut être trouvée dans [2] d'où elles proviennent.

2.5.1 SEBI

Nous allons maintenant aborder la première version de l'algorithme sur lequel nous avons basé notre travail. Il s'agit du SEBI [4], pour Sequential Evolutionary BIclustering. Il a été écrit par les Professeurs Federico Divina et Jesus S. Aguilar-Ruiz. Cet algorithme fait partie de la nouvelle famille d'algorithmes développée par [6] et nous allons en détailler plus amplement le fonctionnement.

En résumé, cette méthode est basée sur les algorithmes génétiques et recherche les biclusters en suivant une stratégie de couverture séquentielle.

Voici une présentation à un haut niveau de l'algorithme [4] :

- Sequential Covering :

```
1   Sequential Covering(Delta)
2   load Expression Matrix EM
3   Repeat
4           EBI(EM,Delta)
5           If (bicluster returned)
6                   store bicluster in Results
7                   adjust weights of EM
8           Else
9                   end_cond met
10          If (max_iter is reached)
11                  end_cond met
12  Until (end_cond is met)
13  Return Results
```

4. Précisons néanmoins que l'implémentation de cet algorithme a nécessité de déterminer divers paramètres ainsi que mettre en place certaines adaptations afin que le programme obtenu fonctionne de manière optimale car la description faite dans [4] est plutôt une présentation de haut niveau qu'un plan détaillé.

- Procédure EBI :

```
1   EBI(Expression Matrix EM,Delta)
2   initialize Population
3   evaluate Population
4   Repeat
5           select parents
6           recombine pairs of parents
7           mutate the resulting offspring
8           evaluate new individuals
9           select individuals for next generation
10  Until (max_iter is reached)
11  best_ind = best individual in Population
12  If (residue(best_ind) < Delta)
13          return best individual in Population
```

L'idée principale de cette stratégie séquentielle consiste à ce que l'algorithme génétique EBI (Evolutionary BIclustering) soit exécuté plusieurs fois. A chacune de ces exécutions, l'algorithme produit le meilleur bicluster possible étant donné les facteurs de taille, row variance, mean squared residue et chevauchement. Si la valeur de mean squared residue du bicluster produit est inférieure à δ, alors il est conservé dans la liste des résultats Results. A chaque fois, l'algorithme garde une trace des éléments constitutifs des biclusters obtenus afin d'utiliser cette information pour minimiser le chevauchement durant les exécutions suivantes de l'algorithme génétique EBI. La stratégie séquentielle s'arrête lorsqu'EBI n'arrive plus à trouver de δ-biclusters ou que l'on en a récolté assez (une centaine).

Le but de EBI est de trouver des δ-biclusters au volume maximum, avec une relativement haute row variance et minimisant le chevauchement entre les biclusters résultats.

Signalons que les individus de l'algorithme génétique représentent les biclusters au moyen de chaînes de bits comme expliqué précédemment.

La première opération réalisée par EBI est l'initialisation de la population. Celle-ci a une taille de 200 individus. Généralement, la population est initialisée totalement au hasard mais ici il a été décidé que les individus initiaux devaient tous représenter des biclusters contenant un seul élément choisi aléatoirement.

Puis, au cours des différentes générations, la population des biclusters évolue au moyen de l'action répétée des opérateurs génétiques de croisement et de mutation. Pour cela, un certain nombre d'individus sont sélectionnés pour devenir parents au moyen de tournois de taille 2. Les parents sélectionnés sont ensuite

combinés au moyen d'opérateurs de croisement appliqués avec une probabilité d'environ 0,85 et les enfants obtenus peuvent par après être mutés avec une probabilité d'environ 0,2. Plus précisément, les différents opérateurs génétiques implémentés dans l'algorithme sont :

- Pour le croisement, on utilise avec des probabilités équivalentes, le "single point crossover", le "two point crossover" et le "uniform crossover". Des deux enfants produits par ces opérateurs, on garde seulement celui ayant la meilleure fitness.
- Pour les mutations, on utilise le "classical mutation operator", un opérateur de mutation qui ajoute des lignes et un autre qui ajoute des colonnes au bicluster.

Le processus continue de la même manière avec la population des nouveaux enfants créés remplaçant totalement la population précédente jusqu'à ce que le nombre maximum de 100 générations soit atteint. Notons que l'on applique aussi à chaque génération l'élitisme avec une probabilité de 0.9, c'est-à-dire qu'on conserve seulement l'individu ayant la meilleure fitness dans la population actuelle pour le passer à la génération suivante. Finalement, à la fin du processus, on sélectionne l'individu ayant la meilleure fitness dans la population finale et étant un δ-bicluster pour le retourner à la procédure séquentielle.

Durant l'algorithme, afin d'évaluer les individus, on va utiliser une fonction de fitness qui agrège en une seule et même fonction le mean squared residue, le volume, la row variance, une fonction de pénalité concernant les chevauchements ainsi que certaines contraintes de poids sur les lignes et colonnes. Il s'agit donc d'un algorithme "single objectif". Précisons que l'objectif final d'EBI est de minimiser cette fonction de fitness.

On notera qu'aucune mesure de distance n'a été incorporée au sein de l'algorithme, seuls les opérateurs de croisement, de mutation ainsi que la procédure d'élitisme permettent d'assurer la diversité de la population.

Pour toute information supplémentaire, le lecteur peut se référer à l'article [4].

2.6 Passage au Multi-objectif

Cette section a été écrite en se basant principalement sur [58].

Quand on recherche des biclusters parmi les données provenant de microarrays, plusieurs objectifs, comme le volume, le mean squared residue, la row variance et d'autres doivent être optimisés en même temps. Souvent ces objectifs sont en conflit les uns avec les autres. Il s'ensuit que ce problème de biclustering peut dès lors être directement vu comme un problème d'optimisation multi-objectif [5]. En effet, lorsqu'il y a deux caractéristiques ou plus en conflit et devant être optimisées, les algorithmes génétiques single objectifs requièrent une formulation appropriée de l'unique fonction de fitness en termes d'une agrégation des différents critères impliqués. Dans de telles situations, les algorithmes évolutionnaires multi-objectifs fournissent une approche alternative plus efficace pour chercher les solutions optimales au problème considéré [3]. C'est ce qui a représenté la motivation majeure pour l'utilisation d'algorithmes génétiques multi-objectifs (MOEAs) afin de trouver les biclusters parmi les données d'expression de gènes [5].

Si nous regardons précisément aux différents objectifs devant être optimisés dans notre cas, il est nécessaire de trouver des biclusters de volume maximum, relativement cohérents et ayant une haute row variance ainsi qu'un faible niveau de chevauchement parmi les biclusters trouvés par l'algorithme. Dans ce contexte, il va falloir optimiser simultanément ces différents objectifs, c'est-à-dire maximiser les premier et troisième objectifs et minimiser les deuxième et quatrième objectifs. Les caractéristiques de ces biclusters, en conflit les unes avec les autres conviennent donc parfaitement pour une modélisation du problème sous forme de multi-objectifs à optimiser [7].

La méthode proposée par [3] a été la première à implémenter un MOEA pour résoudre ce problème de biclustering.

2.6.1 Les algorithmes génétiques multi-objectifs (MOEAs)

Certains problèmes du monde réel venant de nombreuses disciplines ont souvent plusieurs objectifs à optimiser en même temps. De tels problèmes sont appelés "multi-objectifs" et leur trouver une solution implique de trouver le meilleur compromis possible parmi les objectifs à considérer. Traditionnellement, les problèmes d'optimisation multi-objectifs sont traités via une variété de techniques de programmation mathématiques qui ont été développées au fil des ans [67]. Cependant, au cours des dernières années, l'utilisation de méta-heuristiques pour résoudre ce type de problème est devenue de plus en plus populaire (exemple : [66]). Une méta-heuristique est une stratégie de haut niveau permettant l'explo-

ration d'un espace de recherche au moyen de différentes techniques. Les méta-heuristiques combinent une procédure de diversification, c'est-à-dire l'exploration de l'entièreté de l'espace de recherche, ainsi qu'une procédure d'intensification, c'est-à-dire qu'elles exploitent l'expérience accumulée par la recherche. Or les algorithmes génétiques single objectifs et multi-objectifs sont des méta-heuristiques [65].

Les algorithmes génétiques multi-objectifs datent du milieu des années 1980 [42], cependant ils sont devenus vraiment populaires au milieu des années 1990. Aujourd'hui, il est même possible de trouver des MOEAs dans pratiquement toutes les disciplines, y compris la biologie [64].

Les MOEAs sont particulièrement courants à cause de leur facilité d'utilisation et d'implémentation ainsi que grâce au fait qu'ils sont moins sensibles que les techniques de programmation mathématique à la manière d'initialiser la recherche et aux caractéristiques du problème. De plus, comme ils traitent simultanément plusieurs solutions candidates (la population), ils permettent de trouver un ensemble fini de solutions Pareto-optimales en une seule exécution de l'algorithme au lieu d'avoir à réaliser une série d'exécutions successives comme c'est le cas dans les techniques traditionnelles de programmation mathématique [63].

Plus précisément, le problème d'optimisation multi-objectif peut être défini comme le problème de trouver une solution dont les caractéristiques satisfont les contraintes posées par le problème et qui optimisent en même temps les différentes fonctions objectifs définies pour celui-ci. Ces fonctions forment une description mathématique des critères, des objectifs à mesurer, lesquels sont souvent en conflit les uns avec les autres. Par conséquent, le terme "optimiser" signifie trouver une solution dont les différentes valeurs rendent le résultat des fonctions objectifs acceptable pour le preneur de décisions [62].

Les caractéristiques représentent les éléments pour lesquels des valeurs doivent être choisies dans le problème d'optimisation. Dans la plupart de ces problèmes, il y a toujours des restrictions imposées par des caractéristiques de l'environnement ou des ressources disponibles (ex. : limitations physiques, temporelles, etc). Ces restrictions doivent être satisfaites afin de considérer une certaine solution comme acceptable. Toutes ces restrictions sont en général appelées contraintes, et elles décrivent les dépendances existant entre les variables impliquées dans le problème.

En optimisation multi-objectif, le but principal est d'optimiser un ensemble de fonctions objectifs simultanément. Dès lors, dans ce contexte, la notion d'optimum change, car dans ce type de problème, l'objectif est de trouver de bons compromis plutôt qu'une solution unique comme c'est le cas dans les problèmes

single objectifs. La notion d'optimum la plus communément adoptée dans ce cas a été proposée par Francis Ysidro Edgeworth [60] et généralisée plus tard par Vilfredo Pareto [61]. Une solution optimale sera dès lors appelée Pareto-optimale dans un problème d'optimisation multi-objectif.

Une solution est dite Pareto-optimale s'il n'existe pas d'autres solutions valides présentant au moins une diminution pour les différentes valeurs observées dans les critères de celles-ci sans causer d'augmentation simultanée de ces valeurs dans au moins un des autres critères considérés (en supposant que l'on cherche à minimiser). L'optimum de Pareto ne donne presque jamais une solution unique mais plutôt un ensemble de solutions. Ces solutions Pareto-optimales sont dites non dominées.

Le concept d'optimalité, sous-jacent au problème d'optimisation multi-objectif, traite donc un ensemble de solutions et se base lui-même sur le principe de dominance Pareto. Et voici exactement les conditions pour qu'une solution donnée domine une autre solution au sens de Pareto. On illustrera ensuite cette propriété par un exemple avec la figure 2.10.

Lorsqu'il y a M fonctions objectifs, une solution X domine une autre solution Y, si les deux conditions (a) et (b) sont respectées :

(a) La solution X n'est pas pire que Y pour les M fonctions objectifs considérées

(b) La solution X est strictement meilleure que Y pour au moins une des M fonctions objectifs

Quand une solution X domine une autre solution Y alors on peut les classer en leur attribuant un rang de telle sorte que $r_x < r_y$ [3].

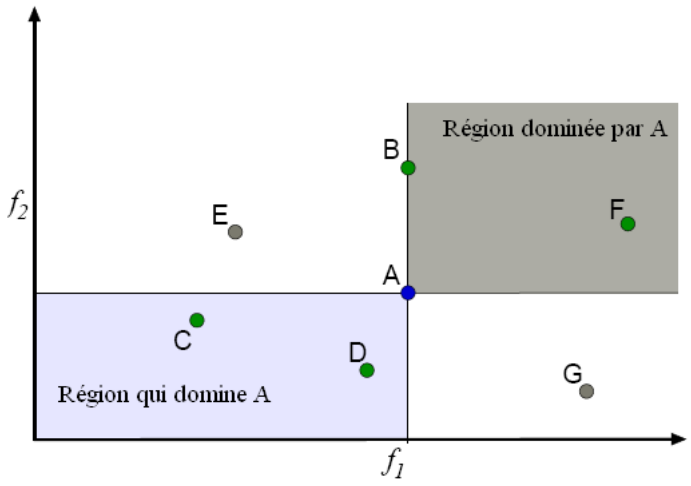


FIGURE 2.10 – Représentation graphique de solutions dominées et de solutions qui dominent un point de référence A pour un problème de minimisation avec deux fonctions objectifs [58]. Les solutions E et G ne sont pas dominées par A, A est meilleure que B et F et moins bonne que C et D mais est à égalité avec E et G.

Si l'on dessine l'espace des fonctions objectifs, l'ensemble de ces solutions Pareto-optimales non dominées est appelé front de Pareto. Les figures 2.11 et 2.12 en donnent deux bonnes illustrations.

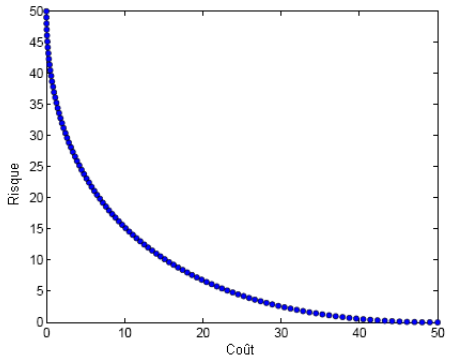


FIGURE 2.11 – Front de Pareto pour un problème d'optimisation hypothétique à deux fonctions objectifs : coût et risque [58].

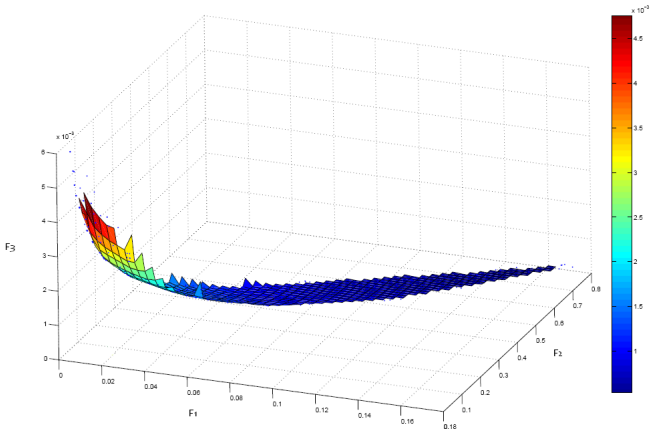


FIGURE 2.12 – Front de Pareto pour un problème d'optimisation hypothétique à trois fonctions objectifs : F_1, F_2 et F_3 [59].

Signalons que bien que les MOEAs génèrent une approximation du front Pareto, en pratique seulement une partie de celui-ci intéresse le preneur de décision pour effectuer son choix.

2.6.2 Détails des composants des MOEAs

Comme nous l'avons vu, en passant d'un algorithme single objectif à un algorithme multi-objectif, c'est surtout la fonction de fitness qui va changer. Les opérateurs de croisement, mutation, les procédures d'initialisation ainsi que de sélection restent identiques. Seule la procédure d'élitisme change aussi.

L'élitisme

Pour les GAs single objectifs, l'élitisme consiste souvent seulement à retenir le meilleur individu de la population à la génération actuelle et de le passer sans le modifier à la génération suivante. En revanche, en cas d'optimisation multi-objectif, l'élitisme implique alors de retenir toutes les solutions non dominées présentes dans la population. Dès lors, au lieu de ne retenir qu'un seul individu, plusieurs vont être gardés. Signalons que cela ajoute des problèmes supplémentaires à prendre en compte (ex. : Doit-on borner le nombre d'individus à retenir ? Si oui, comment choisit-on les individus à conserver ? , etc).

2.6.3 SMOB

Suite aux constatations faites précédemment sur la nature du problème abordé dans le cadre du biclustering de données d'expression et sur ses objectifs conflictuels ainsi que sur la manière de formuler ceux-ci, le travail principal a été de transformer le SEBI en un algorithme génétique multi-objectif tout en gardant les mêmes critères de sélection et d'évaluation des biclusters. A savoir : trouver des δ-biclusters au volume maximum, avec une relativement haute row variance et minimisant le chevauchement entre les biclusters résultats.

Cette transformation a eu comme résultat la seconde version de l'algorithme sur lequel nous avons basé notre travail. Il s'agit du SMOB [5], pour Sequential Multi-Objective Biclustering. Il a également été écrit par les Professeurs Federico Divina et Jesus S. Aguilar-Ruiz[5] et fait aussi partie de la même famille d'algorithmes développée par [6]. Nous allons en détailler plus amplement le fonctionnement.

En résumé, cette méthode est basée sur les algorithmes génétiques et recherche les biclusters en suivant une stratégie de couverture séquentielle. Néanmoins à la différence de sa première version single objectif, celle-ci s'inspire des techniques multi-objectifs.

L'idée principale de la stratégie séquentielle reste la même et consiste à ce que l'algorithme génétique MOEB (Multi-Objective Evolutionary Biclustering) soit exécuté plusieurs fois. A chacune de ces exécutions, l'algorithme produit le meilleur bicluster possible étant donné les facteurs de taille, row variance, mean squared residue et chevauchement. Si la valeur de mean squared residue du bicluster produit est inférieure à δ, alors il est conservé dans la liste des résultats Results. A chaque fois, l'algorithme garde une trace des éléments constitutifs des biclusters obtenus afin d'utiliser cette information pour minimiser le chevauchement durant les exécutions suivantes de l'algorithme génétique MOEB. La stratégie séquentielle s'arrête lorsque le MOEB n'arrive plus à trouver de δ-biclusters ou que l'on en a récolté assez (une centaine).

5. De la même manière que pour le SEBI, l'implémentation de cet algorithme a nécessité de déterminer divers paramètres ainsi que mettre en place certaines adaptations afin que le programme obtenu fonctionne car la description faite dans [5] est plutôt une présentation de haut niveau qu'un plan détaillé.

Signalons que la représentation des individus et donc des biclusters reste inchangée et se fait toujours au moyen de chaînes de bits comme expliqué précédemment.

Ensuite, la première opération réalisée par MOEB est l'initialisation de la population. Celle-ci a toujours une taille de 200 individus mais on détermine cette fois-ci au hasard le nombre de gènes et de colonnes que va contenir chaque individu de la population de sorte que chaque bicluster représenté ait un volume différent. On sélectionne ensuite aléatoirement les différents bits jusqu'à atteindre le nombre nécessaire dans chaque individu en les passant à 1.

Puis, au cours des différentes générations, la population des biclusters évolue au moyen de l'action répétée des opérateurs génétiques de croisement et de mutation. Pour cela, un certain nombre d'individus sont sélectionnés pour devenir parents au moyen de tournois de taille 4 cette fois-ci. Les parents sélectionnés sont ensuite combinés au moyen d'opérateurs de croisement et les enfants obtenus peuvent par après être mutés. Plus précisément, les différents opérateurs génétiques implémentés dans l'algorithme restent les mêmes :

- Pour le croisement, on utilise le "single point crossover", le "two point crossover" et le "uniform crossover", ce dernier ayant la plus grande probabilité d'application des trois opérateurs. Des deux enfants produits par ces opérateurs, on garde seulement celui ayant la meilleure fitness.
- Pour les mutations, on utilise le "classical mutation operator", un opérateur de mutation qui ajoute des lignes et un autre qui ajoute des colonnes au bicluster.

Le processus continue de la même manière avec la population des nouveaux enfants créés remplaçant totalement la population précédente jusqu'à ce que le nombre maximum de 100 générations soit atteint. Notons que l'on applique aussi à chaque génération l'élitisme qui conserve tous les individus non dominés de la population pour les faire passer à la génération suivante. Finalement, à la fin du processus, on sélectionne l'individu ayant la meilleure fitness au sein de la population finale et étant un δ-bicluster pour le retourner à la procédure séquentielle.

Durant l'algorithme, afin d'évaluer les individus, on va utiliser une fonction de fitness qui va cette fois-ci se baser sur la dominance Pareto des individus entre eux au sein de la population. Pour affirmer qu'un individu en domine un autre, on va utiliser les caractéristiques de mean squared residue, de row variance et de volume. C'est pour cela qu'on parle d'un algorithme "multi-objectif". Néanmoins, afin d'assurer la diversité de la population, on va incorporer à cette fonction de fitness deux mesures de distance. Une mesure se basant sur la distance en termes d'objectif (mean squared residue, row variance, etc) et l'autre se basant sur la densité moyenne des individus de la population. Des pénalités peuvent de plus être ajoutées à cette fonction de fitness si l'individu considéré n'est pas un

δ-bicluster valide et s'il chevauche des biclusters présents dans la liste des résultats Results. Ces différents ajouts faits à la fonction de fitness vont transformer l'approche "multi-objectif" initiale en une approche "single objectif". C'est-à-dire que l'on ne va plus se diriger vers un front de Pareto, un ensemble de solutions, mais plutôt vers une seule solution, un seul individu. Cela correspond en effet totalement avec la stratégie séquentielle adoptée par l'algorithme car celle-ci ne conserve au final qu'un seul individu et en chercher un ensemble aurait donc été inutile. Précisons finalement que l'objectif final du MOEB est de minimiser cette fonction de fitness.

Pour toute information supplémentaire, le lecteur peut se référer à l'article [5] .

2.7 Vers un algorithme mémétique

Cette section a été écrite en se basant principalement sur [7], [40], [6] et [3].

Le but des différents algorithmes présentés ci-avant, SEBI (un algorithme génétique single objectif) puis SMOB (l'amélioration du SEBI en multi-objectif), est d'utiliser une approche évolutionnaire afin de trouver des biclusters au sein d'une matrice d'expression de gènes donnée. La dernière version, le SMOB, vise à améliorer encore la méthode initiale développée pour donner les meilleurs compromis entre les objectifs à optimiser. Cependant, les solutions trouvées comprennent encore des biclusters qui ne satisfont pas toutes les contraintes, il est nécessaire de guider la recherche de telle manière que l'ensemble trouvé soit plus satisfaisant. Dans ce contexte, plusieurs améliorations vont être apportées à l'algorithme afin d'orienter au mieux son exploration et de le faire converger plus rapidement vers de meilleures solutions. Dans ce cas, on va alors parler d'algorithme mémétique. Une autre raison avancée pour motiver ces changements est que certains résultats obtenus révèlent que les algorithmes génétiques single et multi objectifs employés seuls obtiennent souvent de "pauvres" biclusters [6].

2.7.1 Les algorithmes mémétiques

Les algorithmes mémétiques (MAs) combinent des méthodes basées sur l'évolution (algorithmes génétiques) avec d'autres techniques connues comme étant bonnes pour la résolution de la classe particulière de problèmes traitée. Ces techniques consistent habituellement en une ou plusieurs phase(s) de recherche locale ou en l'utilisation d'informations spécifiques au problème au sein des opérateurs utilisés dans les méthodes évolutionnaires. La figure 2.13 illustre justement les différentes techniques possibles à ajouter.

Le choix de l'introduction de MAs est motivé par le fait que les algorithmes évolutionnaires sont reconnus bénéfiques pour l'exploration de l'espace de recherche mais moins bons pour l'exploitation de celui-ci. Cela signifie que les algorithmes génétiques peuvent rapidement trouver de bonnes solutions mais qu'ils ne sont néanmoins pas suffisamment bons pour continuer à améliorer ces solutions jusqu'à l'optimum. Ceci est dû en partie à la nature aléatoire des opérateurs (de croisement, de mutation, etc) utilisés dans les algorithmes génétiques.

De plus, dans beaucoup de problèmes où ces algorithmes évolutionnaires sont appliqués, il existe souvent une grande connaissance préalable du problème. Dans de tels cas, on peut tirer un certain profit en utilisant ces informations au sein de

6. On parlera de "pauvres" biclusters si ils présentent de mauvaises valeurs pour les différentes caractéristiques considérées (comme un très haut mean squared residue, une très basse row variance, etc).

l'algorithme génétique, sous la forme d'opérateurs spécialisés, de bonnes idées et pratiques à appliquer, tout en prenant garde à ce que la recherche ne soit pas trop biaisée et empêche donc la génération de solutions intéressantes. On constate en effet que cette combinaison des méthodes évolutionnaires et heuristiques fonctionne souvent mieux que les deux méthodes prises séparément.

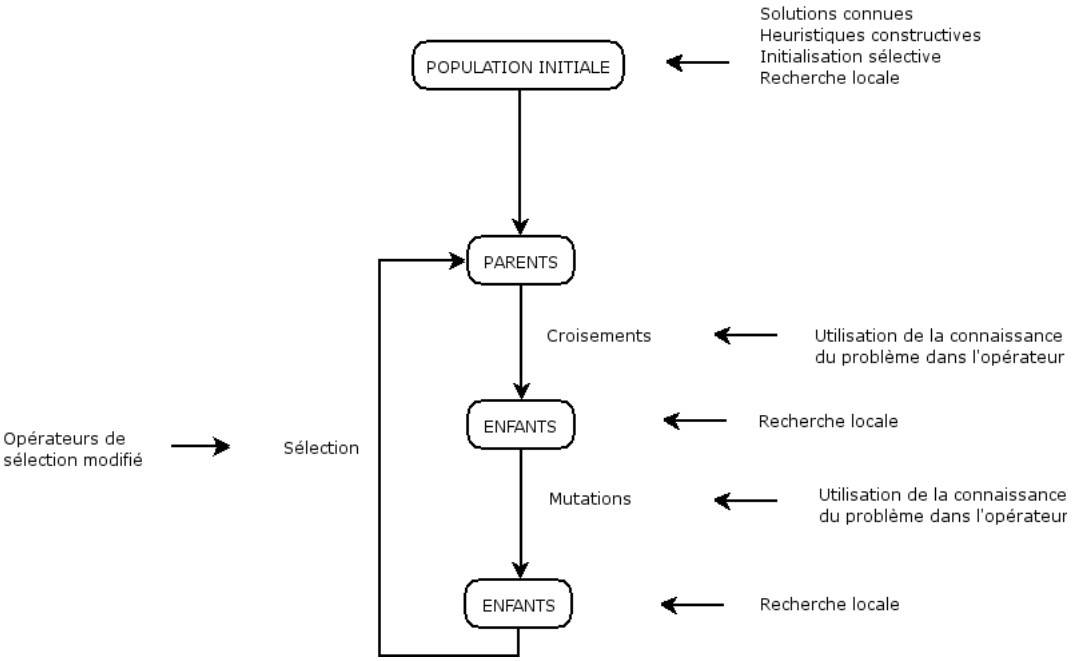


FIGURE 2.13 – Les différentes étapes de l'algorithme génétique où l'on peut incorporer une connaissance du problème et des améliorations.

La première place est celle de l'initialisation de la population. Les algorithmes génétiques initialisent typiquement la population avec un ensemble de solutions candidates prises au hasard. En vue d'améliorer cette étape, on pourrait plutôt penser à initialiser la population avec un ensemble de solutions déjà connues comme étant bonnes, ces solutions étant soit connues à l'avance ou sont le résultat de certaines méthodes de constructions heuristiques permettant de les déterminer. Une méthode de recherche locale pourrait aussi être appliquée afin de générer aléatoirement les solutions de telle manière que la population initiale consiste en un ensemble d'individus localement optimaux. La recherche locale peut être décrite comme un processus itératif examinant les possibilités d'amélioration d'une solution actuelle considérée et essayant de remplacer celle-ci par une version améliorée d'elle-même si elle existe. Une autre voie, appelée initialisation sélective, consiste à générer un grand ensemble de solutions au hasard et d'ensuite sélectionner un sous-ensemble de ces solutions. Initialiser la population

de manière non aléatoire peut avoir des avantages intéressants. En effet, utiliser des solutions existantes évite les efforts de calcul nécessaires à l'obtention de solutions déjà connues. Un autre avantage vient du fait que cette population initiale de solutions choisies va biaiser la recherche et l'emmener vers des régions contenant de bonnes solutions, augmentant ainsi l'efficacité de l'algorithme.

Une autre étape améliorable est celle qui concerne les opérateurs de croisement et de mutation. De nouveaux opérateurs intelligents peuvent être utilisés à la place des opérateurs classiques. Ces opérateurs intelligents exploitent la connaissance et les informations connues sur le problème afin de produire de meilleurs enfants.

Citons aussi la manière la plus commune d'améliorer un algorithme génétique qui est l'application d'une ou plusieurs phase(s) de recherche locale agissant sur les nouveaux individus créés par mutation ou croisement. De cette manière, il y a une chance que si les opérateurs génèrent une solution proche de l'optimum, la solution puisse être améliorée de telle manière que l'optimum soit atteint. Signalons aussi que lorsqu'on applique une procédure de recherche locale, il existe alors deux options concernant la solution donnée suite à son exécution :

- Soit cette solution est seulement utilisée afin de déterminer la valeur objective potentielle de l'individu original, tout en gardant cet individu original inchangé, on parle d'approche Baldwinienne.
- Soit la nouvelle solution remplace l'individu original, on parle d'approche Lamarckienne [6].

Finalement, la connaissance du problème peut aussi être utilisée via l'opérateur de sélection. Elle va pouvoir promouvoir la diversité au sein de la population, évitant de cette manière une convergence prématurée de la population vers une solution non optimale de l'espace de recherche. Cette convergence prématurée est un problème affectant beaucoup de MAs. C'est dû à l'application des méthodes de recherche locale. En effet, si les phases de recherche locale continuent jusqu'à l'obtention d'un optimum local, cela va inévitablement causer une perte de diversité de la population.

2.7.2 Méthode de recherche locale : Cheng & Church

Présentons maintenant une des procédures de recherche locale les plus couramment rencontrées dans les algorithmes mémétiques traitant le problème du biclustering de données d'expression de gènes : l'heuristique de recherche greedy proposée par Cheng & Church.

Le but premier de cet algorithme est de trouver des biclusters respectant certaines contraintes d'homogénéité, de cohérence au sein d'une matrice d'expression, c'est-à-dire trouver des biclusters ayant un mean squared residue plus bas qu'un δ donné. Outre le fait qu'elle respecte parfaitement la description des méthodes de recherche locale donnée auparavant, cette méthode est, de plus, parfaitement adaptée au domaine du problème à traiter. C'est pourquoi elle est autant utilisée. En effet, si l'on se place dans le contexte du biclustering par algorithme génétique expliqué précédemment, cette méthode effectue justement sa recherche en suivant les mêmes objectifs.

Si l'on considère un des biclusters provenant de l'exécution de l'algorithme génétique utilisé, il se peut que celui-ci contienne certains gènes ou conditions non pertinents. Cette situation se rencontre à chaque étape de l'exécution de l'algorithme (initialisation, après les croisements, mutations, etc). Ces gènes et conditions ont besoin d'être supprimés du bicluster. En outre, afin d'en obtenir de meilleurs, certains autres gènes et conditions plus pertinents doivent y être incorporés. C'est justement le principe utilisé par l'algorithme de Cheng & Church.

L'algorithme débute en prenant un bicluster (I, J) et la matrice d'expression correspondante en entrée, il consiste en trois étapes. NB : Dans la suite de l'explication, une "node" fait référence à un gène ou une condition.

1) Dans la première étape, plusieurs nodes sont supprimées à chaque itération. Cette étape est seulement réalisée si le nombre de gènes ou conditions du bicluster est au-dessus d'un certain seuil (de 100 par défaut). Elle consiste à calculer e_{iJ} pour tout $i \in I$, e_{Ij} pour tout $j \in J$, e_{IJ} et r_{IJ}. Si $r_{IJ} \leq \delta$ alors on retourne le bicluster. Ou sinon, on enlève tous les gènes $i \in I$ tels que $\frac{1}{|C|} \sum_{j \in J} (e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2 > \alpha r_{IJ}$ pour α donné.
   On recalcule ensuite toutes les moyennes et on réalise les mêmes opérations avec les conditions. On itère ce processus jusqu'à ce qu'on n'obtienne plus d'amélioration. Signalons que le paramètre α détermine la vitesse de suppression des nodes et que plus la valeur de celui-ci est haute, moins on va en supprimer. On appelle cette étape "Multiple Node Deletion".

2) La seconde étape réalise les suppressions de nodes une à une. A chaque itération, on supprime la node ayant la plus grande valeur de
d(i) = $\frac{1}{|C|}\sum_{j\in J}(e_{ij} - e_{iJ} - e_{Ij} + e_{IJ})^2$
L'équation pour les conditions est analogue. Cette étape est itérée jusqu'au moment où le mean squared residue descend sous le δ donné. On appelle cette étape "Single Node Deletion".

3) Dans l'étape finale, tous les gènes et conditions non contenus dans le bicluster sont testés et si l'un d'entre eux peut être ajouté sans augmentation du mean squared residue, on l'ajoute. Cette étape est itérée jusqu'à ce qu'aucune node supplémentaire ne puisse plus être ajoutée. Notons que nous ne considérons pas ici l'inverse des lignes comme c'est fait dans l'implémentation originale de l'algorithme[7]. On appelle cette étape "Node Addition".

Signalons que cette méthode de recherche locale est tirée de l'algorithme plus général proposé par Cheng & Church dans leur article décrivant les concepts du mean squared residue, de la row variance, etc. Ces derniers ont en effet présenté un algorithme destiné à travailler de manière indépendante d'autres algorithmes. Il se base sur les mêmes idées et consiste à appliquer itérativement les trois étapes décrites ci-dessus de manière à obtenir un nouveau bicluster à chaque exécution de l'algorithme en partant à chaque fois de la matrice expression dans son entièreté. Tout cela en ayant pris soin de cacher les biclusters obtenus précédemment par des nombres aléatoires afin de ne plus les trouver à nouveau.

Pour toute information supplémentaire, le lecteur peut se référer à l'article [2].

7. On ne considère pas l'inverse des lignes car le SEBI [4] et le SMOB [5] ne les utilisent pas non plus ainsi qu'un grand nombre d'autres algorithmes génétiques traitant du même problème [8, 6, 3].

Chapitre 3

Algorithme et implémentation

3.1 Problématique et contexte de l'étude

Ce travail s'intègre dans l'ensemble des autres travaux que nous avons déjà cités ([3, 7, 8, 5, 4, 6]) et qui traitent du même problème de biclustering de données provenant de microarrays via l'une ou l'autre approche plus ou moins similaire à la nôtre. Nous avons déjà eu aussi l'occasion de détailler plus précisément dans le chapitre précédent toute la théorie sur laquelle nous allons baser nos développements. Notons directement que même si les théories entourant le strict domaine des microarrays ne sont pas liées à l'informatique, mais bien à la biologie, l'informatique apporte juste ici dans ce travail une solution sans sortir de son cadre en ne considérant uniquement le problème que sous l'angle d'un traitement numérique de données. On ne s'attardera donc pas sur la validation biologique des résultats trouvés par notre approche, car cela se situe hors de notre champ de compétence et ces résultats pourront faire l'objet d'une analyse détaillée par la suite grâce à des personnes spécialisées dans le domaine. Il s'agit ici d'essayer d'améliorer l'algorithme existant en se basant sur la théorie et les moyens existants afin d'obtenir de meilleurs résultats sur tout un ensemble de caractéristiques en gardant des performances convenables.

Signalons quand même que certains auteurs ont confirmé par des tests que ces approches analysant les données d'expression des gènes via divers algorithmes de biclustering pouvaient conduire à des résultats valables et exploitables en biologie [2, 8, 3]. Pour ce faire, ils ont utilisé des outils permettant de mesurer la pertinence des données biologiques trouvées, comme notamment les outils OntoTools [77] ou GOTermFinder [78]. On citera aussi plus particulièrement l'équipe constituée par Y. Cheng (Professeur en informatique) et G. Church (Professeur en génétique), qui était donc formée par des spécialistes dans les questions biologiques et informatiques et qui ont développé les définitions utilisées dans la

famille d'algorithmes qui nous intéresse ici.

3.2 Améliorations apportées

Afin de résoudre le problème du biclustering de données d'expression de gènes, on a vu au fil du chapitre précédent que la réflexion générale menée dans ce but a évolué étape par étape. Tout d'abord en proposant diverses approches afin de trouver la meilleure solution (iterative row and column clustering combination [21, 22, 23], divide and conquer [20, 24], exhaustive bicluster enumeration [25, 26, 27], distribution parameter identification [28, 29, 30, 31], greedy iterative search [2, 32, 33, 34, 35, 36, 37]). Ensuite, c'est plus particulièrement sur cette dernière famille d'algorithmes que l'on s'est tourné et sur laquelle repose notre travail. En effet, il se base sur la théorie et les concepts donnés par Cheng & Church dans leur article [2]. Plus spécifiquement, la nouvelle approche proposée pour la première fois par [6] repose sur la découverte de biclusters au moyen d'algorithmes génétiques en combinaison avec les idées expliquées dans cet article (mean squared residue, row variance, etc). Suivant cette voie, d'autres auteurs ont aussi développé leurs propres méthodes ([3, 7, 8, 5, 4, 6]). Et comme nous l'avons expliqué précédemment, notre travail se base sur l'une d'elles : le SMOB [5], un algorithme génétique multi-objectif, développé par les Professeurs Federico Divina et Jesus S. Aguilar-Ruiz améliorant eux-mêmes leur propre algorithme, le SEBI [4], un algorithme génétique single objectif.

Comme nous l'avons vu aussi dans le chapitre précédent, une manière d'améliorer encore plus l'efficacité de la méthode proposée consiste à y incorporer des procédures de recherche locale ainsi qu'à utiliser les informations connues spécifiques au problème au sein des opérateurs des méthodes évolutionnaires. Nous avons donc cherché à obtenir ce qu'on appelle un algorithme mémétique, c'est-à-dire l'étape suivante dans le processus de développement du SMOB.

C'est en suivant cette voie que nous avons continué le développement de ce programme afin d'améliorer les résultats obtenus principalement au niveau du mean squared residue (à diminuer), de la row variance (à augmenter), du volume (à augmenter), des chevauchements (à diminuer), de la couverture totale (à augmenter), des lignes et des colonnes (à augmenter) [1] en moyenne pour l'ensemble de tous les biclusters trouvés comme solutions [2], comme on l'avait déjà introduit dans le chapitre précédent dans les sections 2.3.1, 2.5.1 et 2.6.3. Précisons aussi l'ordre d'importance suivi parmi ces caractéristiques dans la recherche des améliorations à apporter, de la plus haute à la plus basse : mean squared residue $\geq$ volume $\geq$ row variance = couverture totale $\geq$ chevauchement = couverture des lignes et des colonnes. Le temps de calcul reste lui aussi à surveiller.

En se basant sur les documents suivants [1, 2, 3, 4, 5, 6, 7, 8] et grâce à l'aide du Professeur Divina nous avons donc cherché les meilleures méthodes permettant d'améliorer cet algorithme. Précisons tout d'abord que l'idée principale de la stratégie séquentielle est restée la même mais que c'est uniquement au niveau de l'algorithme génétique MOEB que l'on a apporté les modifications. Ce qui a finalement conduit à incorporer au sein de ce dernier des procédures de recherche locale et des opérateurs génétiques intelligents [3].

1. La couverture totale est le rapport entre le nombre d'éléments de la matrice expression impliqués dans un des biclusters résultats et le nombre d'éléments total de celle-ci. De même pour la couverture des lignes (colonnes) qui sont le rapport entre le nombre de lignes (colonnes) de la matrice expression impliqué(e)s dans un des biclusters résultats et le nombre de lignes (colonnes) total de celle-ci.

2. De plus, on préférera des biclusters résultats présentant proportionnellement le plus grand nombre de colonnes possibles pour le dataset considéré, alors que le nombre de lignes pourra varier. En effet, en couvrant un maximum de colonnes, on couvre donc un maximum de conditions expérimentales pour lesquelles les gènes associés dans le bicluster évoluent de manière cohérente. Ce qui a donc plus de valeur d'un point de vue biologique.

3. De la même manière que pour le SEBI et le SMOB, l'implémentation de ce nouvel algorithme a nécessité de déterminer divers paramètres ainsi que mettre en place certaines adaptations par rapport à la littérature disponible et aux idées initiales afin que le programme obtenu fonctionne.

En effet, nous avons plus précisément ajouté dans le SMOB les trois éléments suivants (qui seront détaillés dans la section suivante) :

- Une procédure de recherche locale basée sur les algorithmes de Cheng & Church consistant plus particulièrement à exécuter successivement sur chacun des individus de la population obtenue à la dernière génération les étapes de "Multiple Node Deletion", "Single Node Deletion" et "Node Addition". Signalons de plus que l'on adopte ici une approche Lamarckienne. Ensuite, le processus va sélectionner l'individu à renvoyer comme solution. Cependant, à la différence du SMOB original, il va faire cela non pas grâce à la fitness mais plutôt grâce au mean squared residue. L'algorithme va donc choisir l'individu ayant le mean squared residue le plus bas parmi tous ceux de la population.

- Une procédure de recherche locale basée elle aussi sur les algorithmes de Cheng & Church mais qui, à la différence de la précédente, se compose uniquement des étapes de "Multiple Node Deletion" et "Single Node Deletion" que l'on exécute successivement à chaque génération sur chacun des individus de la population après la phase de mutation. Précisons aussi que l'on adopte ici une approche Lamarckienne avec les individus obtenus.

- Des opérateurs génétiques intelligents, c'est-à-dire que l'on a modifié les opérateurs de mutation du SMOB de telle manière que partant d'un individu donné, ceux-ci exécutent un certain nombre de fois l'opérateur classique choisi et conservent le meilleur résultat obtenu par rapport au mean squared residue. Ce qui donne plus précisément pour la version intelligente du "classical mutation operator" :

```
1   Intelligent_classical_mutation_operator(Individu ind)
2   Repeat
3           Application du Classical_mutation_operator sur ind
4   Until(NB_ESSAI atteint)
5   best_ind = individu choisi parmi ceux générés ayant le plus
6                   petit mean squared residue
7   return best_ind
```

  Signalons que les modifications ont été apportées de manière similaire sur l'opérateur de mutation qui ajoute des lignes et sur celui qui ajoute des colonnes au bicluster.

On obtient donc au final l'algorithme suivant :

```
1   MOEB(Expression Matrix EM,Delta)
2   initialize Population
3   evaluate Population
4   Repeat
5           select parents
6           recombine pairs of parents
7           mutate the resulting offspring with intelligent mutation operators
8           local search on individuals
9                   (Multiple Node Deletion + Single Node Deletion)
10          evaluate new individuals
11          select individuals for next generation
12  Until (max_iter is reached)
13  local search on individuals
14          (Multiple Node Deletion + Single Node Deletion + Node Addition)
15  best_ind = best individual in Population according to residue
16  If (residue(best_ind) < Delta)
17          return best individual in Population
```

3.3 Détails sur le choix des améliorations

Précisons tout d'abord que les tests effectués ici pour s'assurer de l'efficacité de l'implémentation proposée ont été effectués en utilisant le Yeast dataset. On trouvera une présentation plus détaillée de la configuration utilisée pour les tests ainsi que des jeux de données utilisés dans le chapitre suivant.

Afin de comparer les effets produits sur les résultats obtenus par les modifications apportées, citons premièrement les résultats d'une exécution du SMOB dans sa version originale sur le Yeast dataset.

	SMOB
Mean squared residue	287,13 (14,81)
Volume	428,68 (202,38)
Row variance	1239,30 (650,99)
Coverage	46,75%
Overlapping	48,64%
Genes coverage	53,67%
Conditions coverage	100,00%
Execution time (sec)	663,00

Précisons premièrement que ce que nous appellerons dorénavant overlapping était appelé auparavant le chevauchement dans les chapitres précédents. De même pour le coverage qui s'appelait la couverture totale et les gènes et conditions coverage qui s'appelaient les couvertures des lignes et des colonnes. Signalons aussi que ce tableau présente les statistiques calculées sur différentes caractéristiques envisagées à partir des résultats obtenus à la suite d'une exécution de l'algorithme. C'est-à-dire que pour le mean squared residue, le volume et la row variance, le premier chiffre indique la moyenne obtenue par calcul sur l'ensemble des biclusters trouvés et le second entre parenthèses indique l'écart type correspondant sur l'ensemble des résultats obtenus. Ensuite, les caractéristiques de coverage, overlapping, gènes coverage et conditions coverage sont des pourcentages. Finalement, l'exécution time mesure un temps en secondes. Ces précisions restent valables pour les tableaux suivants.

Le détail de ces résultats se trouve en annexe C.1.

Apportons maintenant plus de détails et d'explications quant au choix des améliorations apportées.

3.3.1 Procédures de recherche locale

L'idée de base concernant l'amélioration à apporter via les procédures de recherche locale était d'utiliser en l'adaptant l'algorithme de Cheng & Church, ce qui avait déjà été fait avec succès dans d'autres algorithmes [3, 6, 7, 8]. De plus, comme nous l'avons déjà expliqué, le fait d'utiliser des améliorations tirant parti de connaissances spécifiques au problème est d'autant plus bénéfique pour les résultats.

Nous avons donc cherché dans un premier temps à utiliser cette procédure, c'est-à-dire l'application successive des étapes de "multiple node deletion", "single node deletion" et "node addition", sur tous les individus de la population finale obtenue à la dernière génération afin de les améliorer au maximum avant d'en choisir le meilleur élément. Néanmoins, c'est sur ce dernier point, le choix de l'individu à renvoyer, qu'une décision plus particulière a dû être prise. En effet, nous avons hésité entre utiliser une approche Lamarckienne totalement classique où les individus sont remplacés par leur version améliorée et où l'on choisit ensuite le meilleur de ceux-ci dans la population grâce à leur fitness qui a été réévaluée après l'utilisation de la procédure de recherche locale et une autre approche, que nous avons finalement choisie, légèrement différente mais plus performante, où les individus sont toujours remplacés par leur version améliorée mais où l'on choisit le meilleur individu de la population non plus grâce à la fitness qui a été réévaluée mais plutôt uniquement par rapport à la valeur du nouveau mean squared residue trouvé. Il faut aussi signaler que l'on pouvait se permettre plus facilement d'exécuter la procédure de recherche locale dans son entièreté à la fin de l'algorithme vu que le temps de calcul pour cette dernière phase ne venait pas interférer en allongeant de manière trop importante le temps global du programme.

Les chiffres suivants, donnant les résultats obtenus respectivement par les deux alternatives précédentes, confirment le bien-fondé de notre choix.

Voici les résultats pour l'approche Lamarckienne classique.

	SMOB(Recherche locale sur la population finale : Approche Lamarckienne classique)
Mean squared residue	226,86 (26,39)
Volume	864,2 (857,35)
Row variance	1230,70 (752,32)
Coverage	58,09%
Overlapping	63,57%
Genes coverage	64,97%
Conditions coverage	100,00%
Execution time (sec)	1091,00

Voici maintenant les résultats pour l'approche Lamarckienne légèrement modifiée.

	SMOB(Recherche locale sur la population finale : Approche Lamarckienne modifiée)
Mean squared residue	187,18 (41,94)
Volume	1019,11 (983,95)
Row variance	1001,39 (701,89)
Coverage	57,76%
Overlapping	61,53%
Genes coverage	66,88%
Conditions coverage	100,00%
Execution time (sec)	960,00

Les détails de ces résultats se trouvent respectivement en annexe E.1.1 et E.1.2.

Via ces résultats, on voit que l'approche Lamarckienne légèrement modifiée que nous avons préféré à l'approche Lamarckienne classique donne de meilleurs résultats. En effet, via celle-ci on obtient en moyenne de meilleurs mean squared residue et de plus grands volumes pour les biclusters résultats, alors que le seul point négatif est d'obtenir des row variance plus basses. De plus, les autres caractéristiques restent au même niveau, en moyenne, pour les deux approches considérées, que ce soit le coverage, l'overlapping, les gènes et conditions coverage ainsi que le temps de calcul global. Ce qui conforte donc notre choix.

Si l'on considère maintenant, l'amélioration des résultats obtenue via l'ajout de cette procédure de recherche locale sur la population finale par rapport aux résultats du SMOB classique, on observe qu'elle touche toutes les caractéristiques

envisagées. En effet, on voit qu'en moyenne, les volumes obtenus sont particulièrement augmentés, de même pour les mean squared residue qui deviennent plus petits, le coverage qui devient plus important, le gènes coverage qui s'améliore aussi, le conditions coverage qui reste stable. Seules certaines caractéristiques deviennent moins bonnes, en particulier la row variance qui diminue fortement, mais aussi l'overlapping qui augmente, même si cette augmentation reste compréhensible au vu des plus grands volumes obtenus. Il reste enfin le temps qui augmente aussi logiquement. Au final, on constate donc que l'utilisation de cette méthode de recherche locale est positive pour les résultats, car elle améliore favorablement les deux caractéristiques les plus importantes pour le problème qui nous occupe, le mean squared residue et le volume. La figure 3.1 en page suivante illustre graphiquement le résultat obtenu par cette amélioration par rapport à la version classique de l'algorithme.

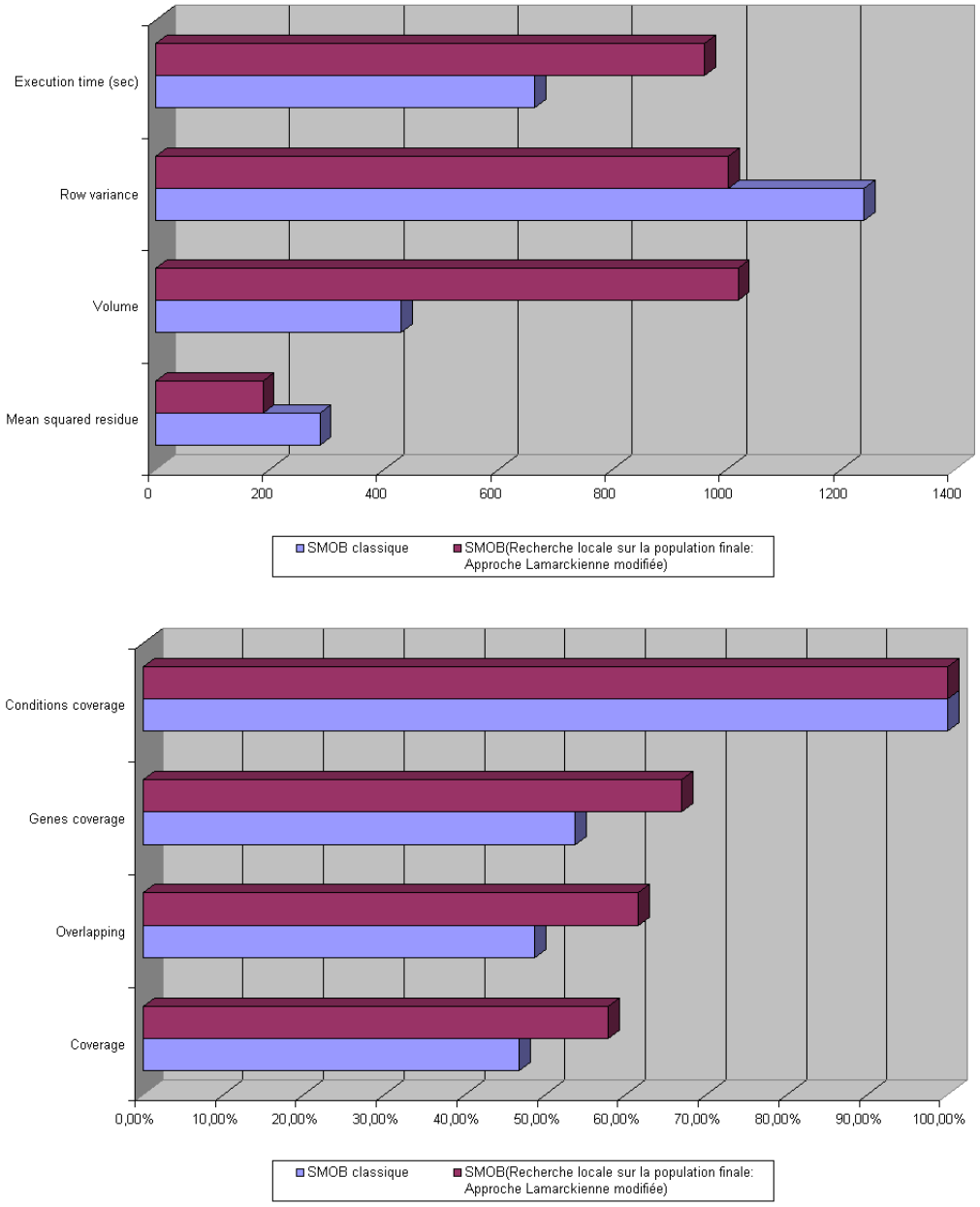


FIGURE 3.1 – Comparaison : SMOB classique avec l'utilisation de la méthode de recherche locale de Cheng & Church ("multiple node deletion", "single node deletion" et "node addition") sur la population finale avec l'approche Lamarckienne modifiée par rapport au SMOB classique.

Dans un second temps, nous avons aussi voulu tester la possibilité d'intégrer la même méthode de recherche locale venant de Cheng & Church sur les individus de la population après avoir appliqué l'opérateur de mutation à chacune des générations. Cependant, après plusieurs essais de diverses combinaisons infructueuses, nous avons bien vite dû nous résoudre à supprimer l'étape de "node addition" de la procédure, celle-ci amenant par la suite, l'algorithme génétique général à traiter une population de biclusters d'une taille beaucoup plus conséquente et rendant donc le temps de calcul global trop important.

Nous obtenons donc les chiffres suivants pour la procédure de recherche locale comprenant uniquement les étapes de "Multiple Node Deletion" et "Single Node Deletion" que l'on exécute successivement à chaque génération sur chacun des individus de la population après la phase de mutation en adoptant une approche Lamarckienne classique avec les individus obtenus.

	SMOB(Recherche locale à chaque génération : Approche Lamarckienne modifiée)
Mean squared residue	278,61 (24,78)
Volume	436,65 (208,60)
Row variance	1225,35 (553,84)
Coverage	55,91%
Overlapping	39,68%
Genes coverage	75,52%
Conditions coverage	100,00%
Execution time (sec)	858,00

Le détail de ces résultats se trouve en annexe E.1.3.

Via ces résultats, on voit effectivement une amélioration des résultats obtenue via l'ajout de cette procédure de recherche locale exécutée sur la population à chacune des générations par rapport aux résultats du SMOB classique. En effet, même si, en moyenne, les caractéristiques de mean squared residue, volume et row variance restent les mêmes, les autres points s'améliorent. C'est-à-dire que le coverage et le gènes coverage augmentent, l'overlapping diminue et le conditions coverage reste stable. Au final, on constate donc que l'utilisation de cette méthode de recherche locale est positive pour les résultats, car elle arrive à améliorer favorablement des caractéristiques fort importantes pour le problème qui nous occupe tout en maintenant les autres caractéristiques de mean squared residue, volume et row variance à leurs niveaux initiaux. La figure 3.2 suivante illustre graphiquement le résultat obtenu par cette amélioration par rapport à la version classique de l'algorithme.

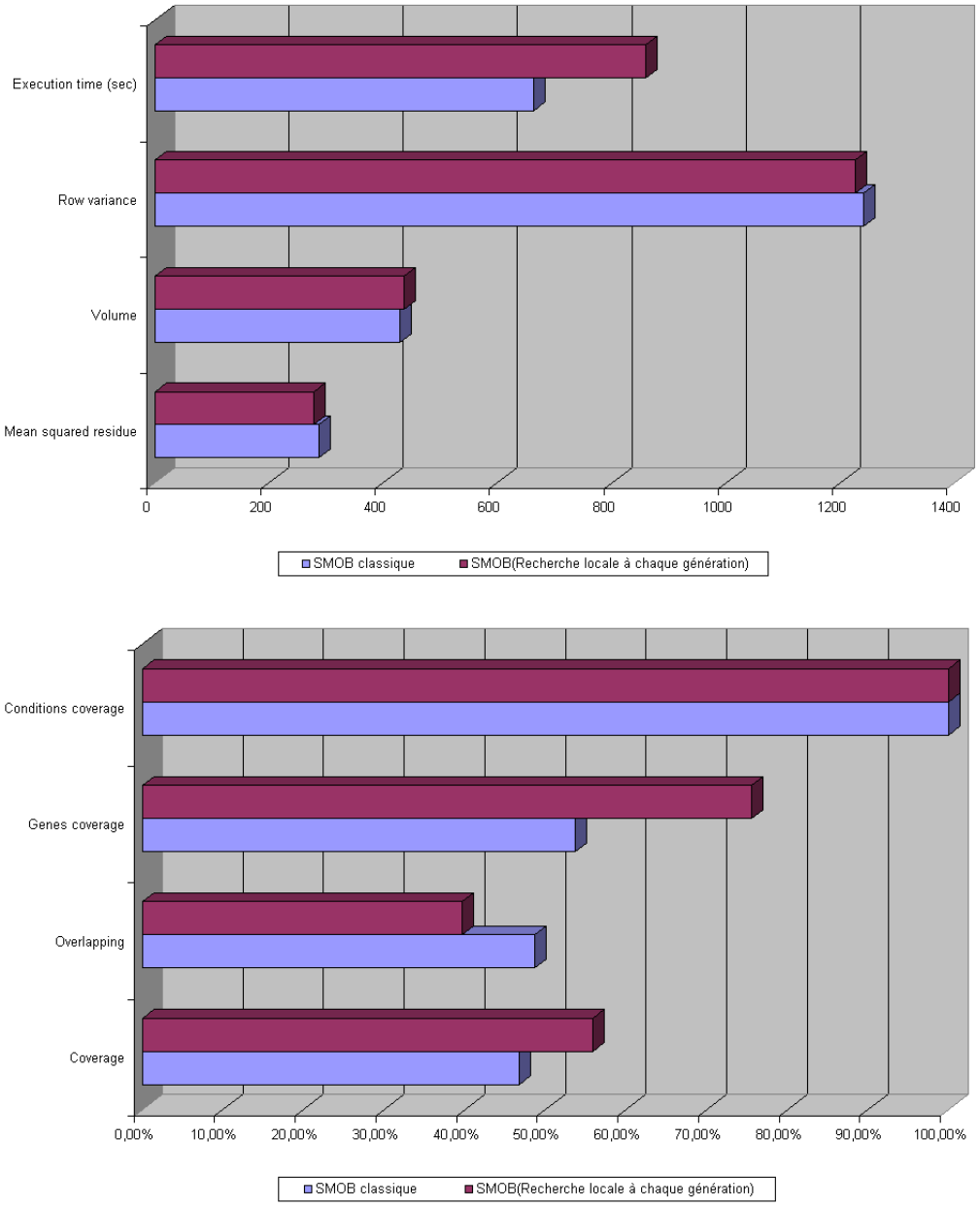


FIGURE 3.2 – Comparaison : SMOB classique avec l'utilisation de la méthode de recherche locale de Cheng & Church ("multiple node deletion" et "single node deletion") à chaque génération avec l'approche Lamarckienne classique par rapport au SMOB classique.

Signalons que l'on peut aussi se demander pourquoi avoir choisi une approche Lamarckienne plutôt qu'une approche Baldwinienne dans les deux cas d'utilisation de la recherche locale présentés ci-dessus. On peut en fait expliquer ce qui a guidé notre choix. Tout d'abord, le fait que l'approche Lamarckienne était conseillée par [6] pour une question de performance au niveau du temps et en gardant une bonne qualité dans les biclusters trouvés, par rapport à l'approche Baldwinienne. Cette approche donnait donc quand même de relativement bons résultats face aux problèmes traités dans [6]. Comme nous nous trouvions plus ou moins dans le même cas, c'est-à-dire celui de l'utilisation combinée d'un algorithme génétique avec des procédures de recherche locale inspirées de Cheng & Church, garder la même approche Lamarckienne semblait une solution convenable. On peut de plus citer quelques autres approches ayant fait ce même choix : [3] et [59].

3.3.2 Opérateurs génétiques intelligents

Finalement, abordons la dernière amélioration consistant à intégrer des opérateurs génétiques intelligents au sein de l'algorithme. La première idée trouvée afin de rendre plus intelligents les opérateurs de mutation consistait à exécuter l'opérateur classique choisi en partant d'un individu donné le nombre de fois nécessaire à ce que le nouvel individu obtenu représente une amélioration significative par rapport au mean squared residue (c'est-à-dire obtenir un mean squared residue plus petit que celui du précédent). Néanmoins, même si cette approche fonctionnait correctement combinée avec le "classical mutation operator", celle-ci était totalement inefficace en combinaison avec les opérateurs de mutation qui ajoutent des lignes et des colonnes. En effet, le temps nécessaire pour trouver une solution acceptable était beaucoup trop long car il est très difficile de réduire le mean squared residue en ajoutant une ligne ou une colonne au bicluster et encore plus en le faisant de manière aléatoire. C'est pourquoi, une version différente a finalement été choisie pour tous les opérateurs de mutation. Elle consistait à exécuter un certain nombre de fois l'opérateur classique choisi en partant d'un individu donné et conserver uniquement le meilleur résultat obtenu par rapport au mean squared residue. Il ne restait alors plus qu'à déterminer au mieux ce nombre d'essais à effectuer. Des tests furent réalisés avec respectivement 25, 50 et 100 essais et c'est finalement le nombre de 25 qui fut retenu car représentant le meilleur compromis par rapport aux caractéristiques considérées.

	SMOB(25 essais)	SMOB(50 essais)	SMOB(100 essais)
Mean squared residue	219,99 (53,36)	199,65 (60,08)	187,71 (59,98)
Volume	553,2 (173,53)	448,02 (150,56)	357,78 (132,82)
Row variance	1348,15 (763.30)	1637,78 (1372,46)	1724,71 (1250,89)
Coverage	55,10%	49,59%	43,46%
Overlapping	53,92%	48,17%	42,61%
Genes coverage	69,34%	66,53%	61,58%
Conditions coverage	100,00%	100,00%	100,00%
Execution time (sec)	1281,00	1672,00	2487,00

Résultats obtenus en combinant le SMOB classique et les opérateurs de mutation intelligents faisant respectivement 25, 50 et 100 essais.
Les détails de ceux-ci se trouvent respectivement en annexe E.1.4, E.1.5 et E.1.6.

Précisons que c'est l'ordre d'importance des caractéristiques des résultats obtenus cité plus haut qui a dicté le choix de conserver le meilleur individu en se basant sur le mean squared residue plutôt qu'une des autres caractéristiques. Néanmoins, l'opérateur de mutation qui ajoute des lignes et celui qui ajoute des colonnes ont par nature un impact positif sur le volume, deuxième caractéristique à prendre principalement en compte. C'est pourquoi nous ne nous sommes pas d'avantage focalisés sur celle-ci au travers de ce nouvel opérateur.

Signalons finalement que cette approche basée sur des opérateurs génétiques intelligents fonctionne correctement et répond parfaitement aux attentes en améliorant globalement les résultats obtenus via le SMOB classique. En effet, on rejette premièrement la version utilisant 100 essais car même si on obtient les meilleurs résultats en termes de mean squared residue, row variance et overlapping parmi les trois tentatives, cela se fait au détriment des performances de coverage, volume et temps qui deviennent pires que celles du SMOB classique. On préférera finalement la version 25 essais à la version 50 essais car en comparaison, celle-ci donne de meilleures performances pour les caractéristiques de volume, coverage, gènes coverage et temps tout en gardant de relativement bons résultats pour le mean squared residue, l'overlapping et la row variance. La figure 3.3 suivante illustre graphiquement le résultat obtenu par cette amélioration par rapport à la version classique de l'algorithme.

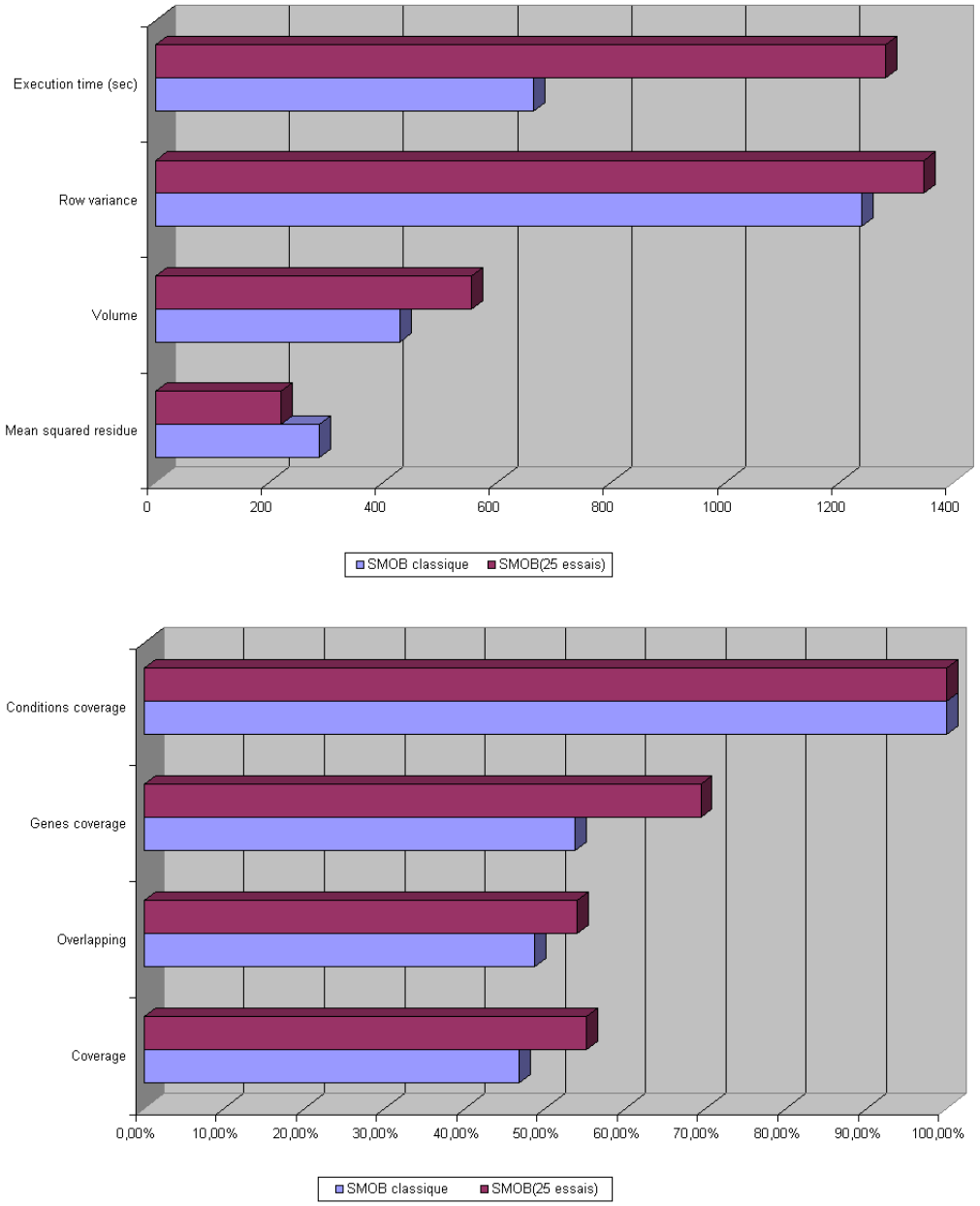


FIGURE 3.3 – Comparaison : SMOB classique avec les opérateurs de mutation intelligents (25 essais) par rapport au SMOB classique.

3.3.3 Combinaison des améliorations

Voici finalement les résultats obtenus en combinant le SMOB classique avec les améliorations expliquées précédemment. Les détails de ceux-ci se trouvent en annexe D.1.

	SMOB	SMOB avec améliorations
Mean squared residue	287,13 (14,81)	196,69 (50,50)
Volume	428,68 (202,38)	802,81 (781,83)
Row variance	1239,30 (650,99)	1412,03 (1029,85)
Coverage	46,75%	69,27%
Overlapping	48,64%	58,32%
Genes coverage	53,67%	90,08%
Conditions coverage	100,00%	100,00%
Execution time (sec)	663,00	1633,00

Au vu des résultats, on peut donc conclure que cette approche, consistant à intégrer les trois améliorations citées au sein de l'algorithme original, fonctionne correctement et répond parfaitement aux attentes en améliorant globalement le SMOB classique. On peut aussi le voir en analysant la figure 3.4 qui illustre justement les résultats obtenus par ces améliorations par rapport à la version classique de l'algorithme. En effet, on obtient bien de meilleurs résultats par rapport à la version classique de l'algorithme mais aussi par rapport aux différentes améliorations prises séparément ou en combinaison deux à deux, leur combinaison globale apporte donc effectivement un bénéfice réel. Ainsi, quelle que soit la caractéristique prise en considération, on obtient de meilleures performances par rapport au SMOB classique (à l'exception du temps de calcul et de l'overlapping qui augmentent néanmoins). Précisons qu'une analyse plus détaillée de ces résultats sera présentée dans la section suivante et ajoutons déjà que les autres jeux de tests disponibles conduisent à la même conclusion positive quant aux performances de cette nouvelle approche employée.

Comme on l'a constaté précédemment, la première amélioration (l'exécution d'une procédure de recherche locale sur les individus de la population finale de la dernière génération) agit positivement sur les caractéristiques suivantes : le mean squared residue, le volume, le coverage et le gènes coverage. Ensuite, la seconde amélioration (l'exécution d'une procédure de recherche locale sur les individus de la population à chacune des générations) agit de manière positive en particulier sur : le coverage, l'overlapping et le gènes coverage. Et finalement, la dernière amélioration (l'utilisation d'opérateurs de mutation intelligents) a un effet positif sur presque toutes les caractéristiques envisagées, à l'exception de l'overlapping. Finalement,on remarque que les temps de calcul augmentent avec chacune des améliorations apportées et les conditions coverage restent stables pour chacune de celles-ci. On voit donc au final que chacun des effets bénéfiques apportés par les améliorations décrites ci-dessus continue à jouer son rôle lorsqu'on les combine

entre elles et donnent donc le résultat final positif que nous avons décrit. Ces différentes améliorations permettent ainsi de mieux raffiner les résultats, c'est-à-dire que l'on va essayer d'optimiser au maximum les individus de la population, ce qui va aller dans le sens de la recherche d'une plus forte sélection et de la création de meilleurs individus plutôt que vers la recherche absolue de diversité et d'un ensemble de solutions. Cette approche coïncide donc bien avec le caractère single objectif observé du SMOB.

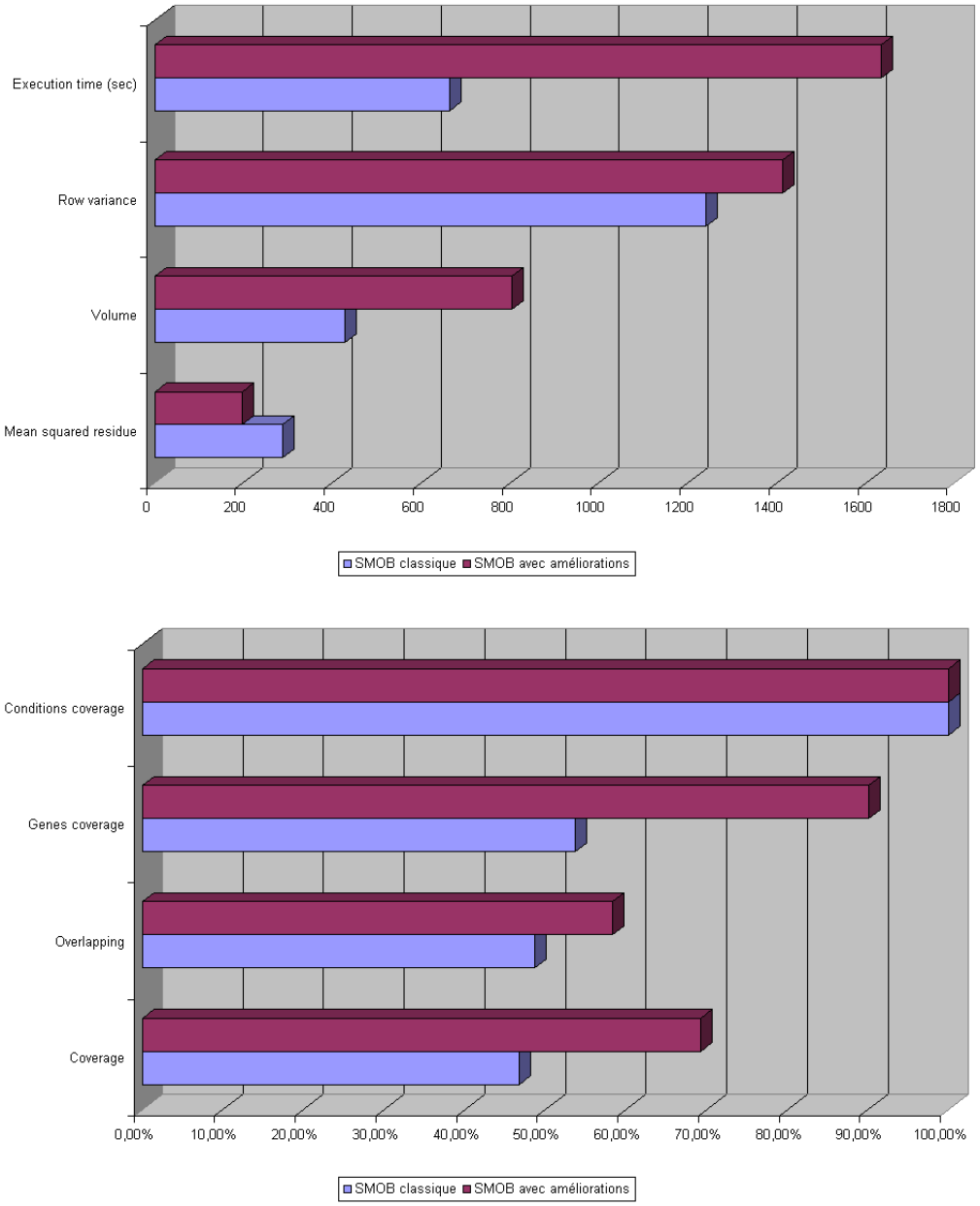


FIGURE 3.4 – Comparaison : SMOB classique avec toutes les améliorations apportées par rapport au SMOB classique.

Chapitre 4

Analyse des résultats

4.1 Configuration de test

L'algorithme présenté a été écrit en Java (JRE 6), tous les tests exécutés et les temps indiqués par la suite ont été mesurés sur un Intel Core 2 Duo P8400 à 2,26 GHz avec 2 GB de RAM.

4.2 Présentation des données utilisées

Cette section a été écrite en se basant principalement sur [2] et [5].

Afin de déterminer la qualité de la méthode proposée précédemment, des tests ont été réalisés en utilisant trois jeux de données (datasets) bien connus. Nous les présentons plus précisément dans cette section en donnant leurs différentes caractéristiques. Néanmoins, au vu des informations disponibles dans la littérature, ces détails restent incomplets.

Le premier dataset utilise les données provenant de la mesure par microarray du cycle cellulaire [1] de la levure Saccharomyces Cervisiae. Il s'agit d'un micro-organisme, une levure utilisée depuis l'aube de l'humanité dans l'élaboration du pain, des yaourts, du vin et de la bière [73]. La matrice d'expression contenue dans ce dataset est de 2884 gènes et 17 conditions. Elle provient de [68]. Ces gènes ont été sélectionnés dans une matrice plus importante contenant 8224 gènes et 17 conditions selon [70] (avec -1 indiquant les valeurs manquantes [74]). Tous ces nombres ont ensuite été transformés et chacune des valeurs originales x de la matrice a été remplacée, via une méthode mise au point dans [2], par $x \rightarrow 100 \log(10^5 x)$, on normalise la matrice. Le résultat est une matrice d'entiers compris entre 0 et 600. Notons que la transformation n'affecte pas les valeurs 0 et -1 présentes dans la matrice.

1. Le cycle cellulaire est l'ensemble des phases par lesquelles une cellule passe entre deux divisions successives [72].

Le second jeu de données concerne les données d'expression du lymphome humain touchant les "B-cells", c'est-à-dire les lymphocytes. Ces données proviennent de [69]. Elles consistent en une matrice d'expression de 4026 gènes et 96 conditions. Chacune des valeurs de la matrice va aussi être remplacée au moyen d'une transformation logarithmique et d'une mise à l'échelle d'un facteur 100. On obtient finalement une matrice d'entiers compris entre -750 et 650, avec 47639 éléments manquants (signalés par la valeur 999 [74]), ce qui représente 12,3 % de la matrice totale.

Le troisième jeu de données est celui du cancer du colon. Ce dataset provient de [71] et contient une matrice d'expression formée de 2000 gènes et de 62 conditions. Ces conditions expérimentales représentent en fait 22 tissus sains et 40 tissus tumoraux provenant du colon. L'analyse se focalise plus précisément sur 2000 gènes parmi les 6500 gènes possibles détectés dans ces tissus [1]. Pour ce troisième jeu de données, chacune des valeurs originales x de la matrice va aussi être remplacée de la même manière par $x \rightarrow 100 \log (10^5 x)$.

Ensuite, les valeurs manquantes de ces différentes matrices vont être remplacées par des nombres entiers choisis aléatoirement. Signalons que bien que ces remplacements de valeurs soient faits de telle manière qu'ils ne forment pas d'ensembles reconnaissables, le risque que ceux-ci apparaissent existe et pourrait affecter la découverte des biclusters. Les nombres aléatoires choisis pour remplacer les valeurs manquantes dans les données provenant de la levure ont été générés afin qu'ils forment une distribution uniforme comprise entre 0 et 800. Pour les données du lymphome humain, la distribution uniforme prend ses valeurs entre -800 et 800.

Les matrices de la levure et du lymphome présentées ci-dessus peuvent être téléchargées à l'adresse suivante [74] (la normalisation des données est déjà réalisée mais pas le remplissage des valeurs manquantes). La matrice du cancer du colon peut aussi être téléchargée en suivant cette adresse [75] (la normalisation des données et le remplissage des valeurs manquantes ne sont pas encore réalisés).

4.3 Présentation des paramètres utilisés

Voici maintenant un tableau récapitulatif des paramètres nécessaires au bon fonctionnement du SMOB classique et qui ont été utilisés pour l'ensemble des différents tests effectués sur les datasets[2].

Nombre de résultats total désirés	100
Taille de la population	200
Nombre de générations	100
Probabilité d'application de l'élitisme	90%
Nombre d'individus conservés par l'élitisme	25
Probabilité d'application du crossover	85%
Probabilité d'application du single point crossover	6,6%
Probabilité d'application du two point crossover	13,3%
Probabilité d'application du uniform crossover	80%
Probabilité d'application de la mutation	20%
Probabilité d'application du classical mutation operator	75%
Probabilité d'application de l'opérateur de mutation ajoutant des lignes	12,5%
Probabilité d'application de l'opérateur de mutation ajoutant des colonnes	12,5%
Taille des tournois de sélection	4

Ensuite, voici les paramètres nécessaires au fonctionnement des améliorations apportées. Encore une fois, ceux-ci ont été utilisés pour l'ensemble des différents tests effectués sur les datasets[2].

α	1,2
Nombre minimum de lignes et de colonnes pour ne pas exécuter l'étape de "multiple node deletion"	100
Nombre d'essais pour les opérateurs de mutation intelligents	25

Signalons que les valeurs d'α et du nombre minimum de lignes et de colonnes pour ne pas exécuter l'étape de "multiple node deletion" concernent les méthodes de recherche locale et sont identiques aux valeurs conseillées dans leur article par Cheng & Church [2]. Le nombre d'essais pour les opérateurs de mutation intelligents a été déterminé par tests comme expliqué précédemment.

2. Tous ces concepts ont été introduits au cours des chapitres précédents.

4.3.1 Détermination des δ

Comme nous l'avons vu précédemment, le seuil δ représente la dissimilarité maximale acceptable entre les éléments du bicluster [7] et même s'il n'existe pas de règles clairement définies et disponibles dans la littérature concernant le choix du paramètre δ pour chaque dataset [3], nous avons fixé les valeurs suivantes :

	δ
Yeast dataset	300
Human Lymphoma dataset	1200
Colon Cancer dataset	500

Ces valeurs de δ généralement admises dans la littérature [2, 3, 4, 5, 6, 7, 8], permettent de plus une comparaison simple entre les résultats obtenus par les différentes approches existantes et futures que nous pouvons rencontrer [3]. C'est plus particulièrement grâce à la méthode suivante décrite en détail dans [2] que l'on a pu estimer premièrement une valeur de δ pour le Yeast dataset en se basant sur des observations existantes provenant de [70] ainsi que grâce aux résultats d'une étude statistique sur les valeurs de mean squared residue rencontrées dans ce dataset. Les clusters reportés dans l'article de [70] avaient des scores compris entre 261 (cluster 3) et 996 (cluster 7), avec une médiane de 630 (cluster 8 et 14). Une valeur de δ de 300 proche des plus basses valeurs rencontrées permet donc de détecter les biclusters les plus homogènes, les plus cohérents. Ensuite, une fois ce premier résultat obtenu, on va l'utiliser pour déterminer la valeur de δ du Human Lymphoma dataset en comparant les caractéristiques des deux matrices expression. Une valeur de 1200 a donc été choisie ici car les valeurs de mean squared residue rencontrées dans ce dernier dataset étaient comprises dans un intervalle deux fois plus grand et présentaient une variance quatre fois plus grande par rapport au Yeast dataset [2]. C'est donc comme cela que nous avons pu déterminer les valeurs de ce paramètre. Comme nous venons de le voir, nous avons donc pu fixer la valeur du paramètre δ à 300 pour le Yeast dataset et à 1200 pour le Human Lymphoma dataset. Ces valeurs trouvées pour les deux premiers datasets proviennent de l'article de Cheng & Church [2] qui se basaient eux-mêmes sur les travaux de [70]. C'est en suivant un raisonnement similaire à celui présenté ci-dessus que la valeur de δ du troisième dataset, le Colon Cancer dataset, a pu aussi être déterminée dans l'article de [5]. En suivant cette méthode, on a donc pu fixer la valeur de δ à 500 étant donné que la taille de sa matrice expression était à peu près le double de celle de la matrice du Yeast dataset [5].

4.4 Yeast dataset

4.4.1 Résultats SMOB classique

Voici les résultats obtenus pour une exécution du SMOB dans sa version originale sur le Yeast dataset. Les détails de ceux-ci se trouvent en annexe C.1.

	SMOB
Mean squared residue	287,13 (14,81)
Volume	428,68 (202,38)
Row variance	1239,30 (650,99)
Coverage	46,75%
Overlapping	48,64%
Genes coverage	53,67%
Conditions coverage	100,00%
Execution time (sec)	663,00

4.4.2 Résultats SMOB avec améliorations

Voici les résultats obtenus en combinant le SMOB classique avec les améliorations pour le Yeast dataset. Les détails de ceux-ci se trouvent en annexe D.1.

	SMOB avec améliorations
Mean squared residue	196,69 (50,50)
Volume	802,81 (781,83)
Row variance	1412,03 (1029,85)
Coverage	69,27%
Overlapping	58,32%
Genes coverage	90,08%
Conditions coverage	100,00%
Execution time (sec)	1633,00

4.4.3 Analyse

Au vu des résultats, on peut donc conclure que cette approche, consistant à intégrer les trois améliorations citées au sein de l'algorithme original, fonctionne correctement et répond parfaitement aux attentes en améliorant globalement le SMOB classique. On peut notamment le voir en analysant la figure 4.1 qui illustre justement les résultats obtenus par ces améliorations par rapport à la version classique de l'algorithme [3]. En effet, quelle que soit la caractéristique prise en considération, on obtient de meilleures performances par rapport au SMOB classique (à l'exception du temps de calcul et de l'overlapping qui augmentent néanmoins). L'augmentation du temps est causée par les améliorations apportées qui nécessitent la réalisation d'un grand nombre de calculs supplémentaires. Concernant l'augmentation de l'overlapping, on peut dire que celle-ci reste raisonnable compte tenu de la forte augmentation du volume des biclusters résultats et du pourcentage de gènes coverage.

Un ensemble de résultats (provenant des résultats cités en annexe D.1) particulièrement intéressants pour leurs caractéristiques et qui a été trouvé par la nouvelle version du SMOB pour le Yeast dataset se trouve en annexe F.1.

3. Il s'agit de la même figure que celle présentée dans le chapitre précédent qui parlait de l'implémentation du nouvel algorithme, la figure 3.4. Nous l'avons copiée ici pour garder la cohérence globale de notre présentation.

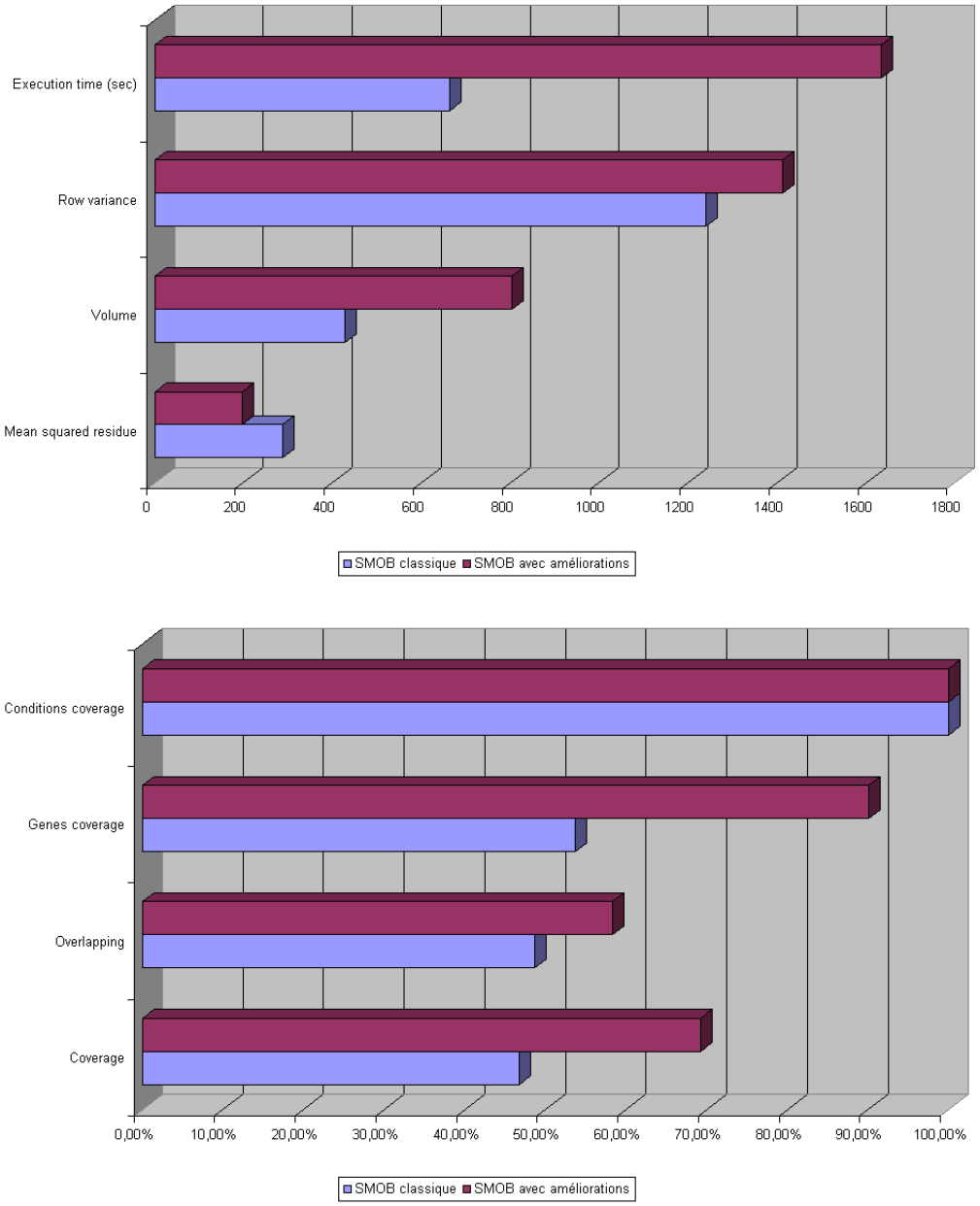


FIGURE 4.1 – Comparaison : SMOB classique avec toutes les améliorations apportées par rapport au SMOB classique pour le Yeast dataset.

La figure 4.2 illustrant la répartition des résultats obtenus par rapport aux deux caractéristiques des biclusters à considérer en priorité (le mean squared residue et le volume) pour le SMOB classique et le SMOB classique avec toutes les améliorations apportées prouve elle aussi l'effet bénéfique des améliorations apportées. En effet, on observe que l'ensemble des résultats obtenus par la nouvelle version de l'algorithme se répartit d'une très bonne manière en ayant certains résultats avec de plus grands volumes et globalement de plus petits mean squared residue.

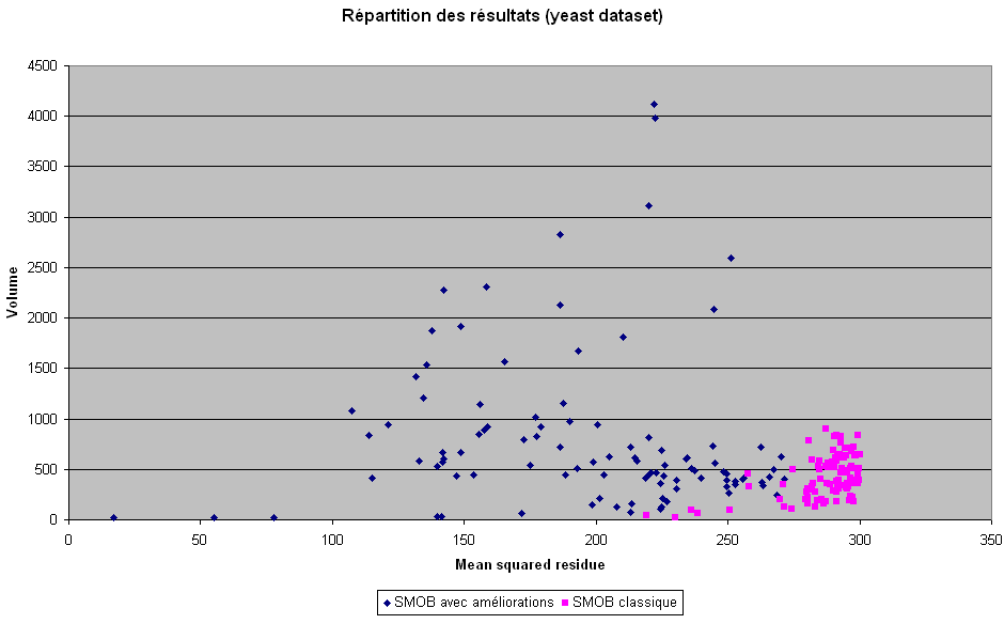


FIGURE 4.2 – Répartition des biclusters résultats obtenus par rapport aux deux caractéristiques à considérer en priorité (le mean squared residue et le volume) pour le SMOB classique et le SMOB classique avec toutes les améliorations apportées pour le Yeast dataset.

En observant les figures 4.3 et 4.4, on se rend aussi compte des bénéfices que les améliorations apportent concernant l'évolution du coverage et de l'overlapping si l'on prend en compte l'ensemble des résultats obtenus et la progression de leurs apports respectifs à ces caractéristiques. Comme constaté dans les résultats précédents, on remarque que la nouvelle approche fournit un meilleur coverage global, même si l'overlapping devient légèrement plus important qu'auparavant.

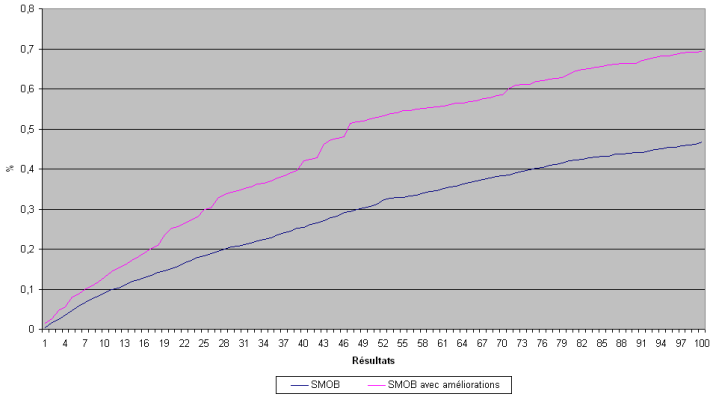


FIGURE 4.3 – Evolution du coverage pour l'ensemble des résultats obtenus pour le Yeast dataset.

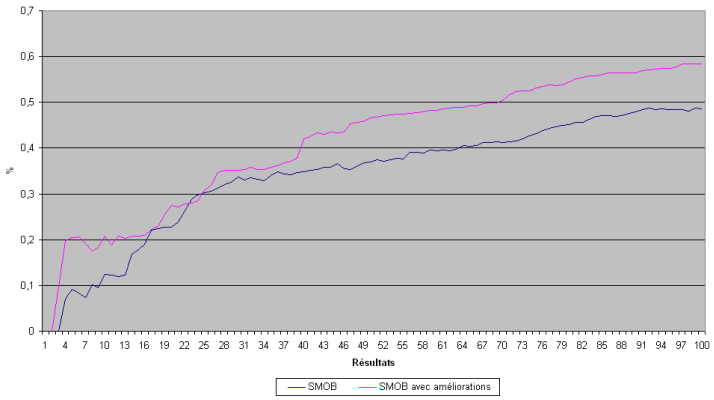
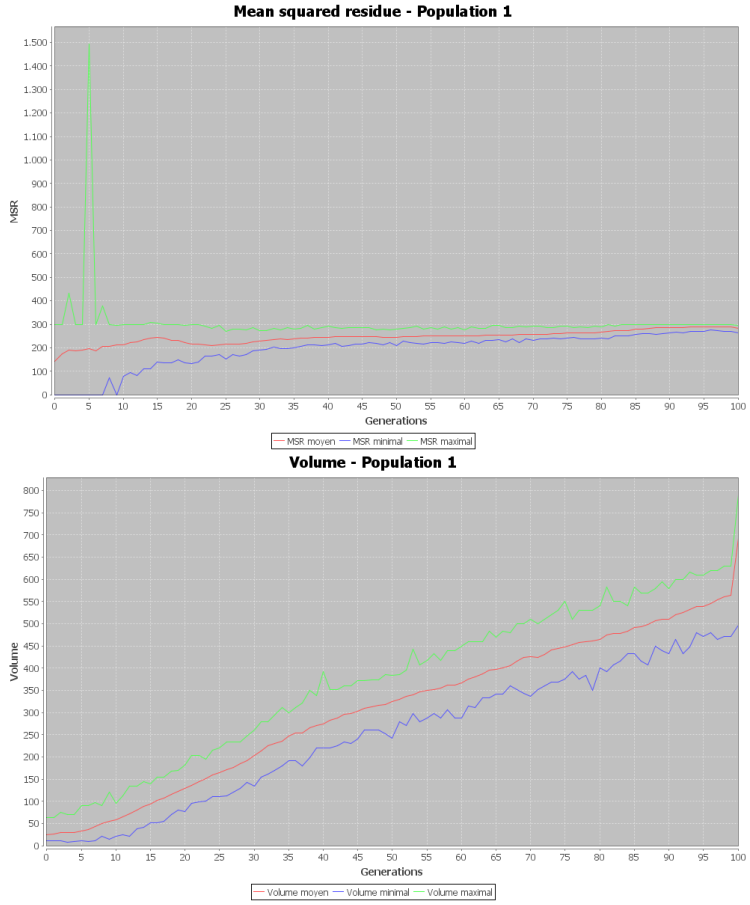
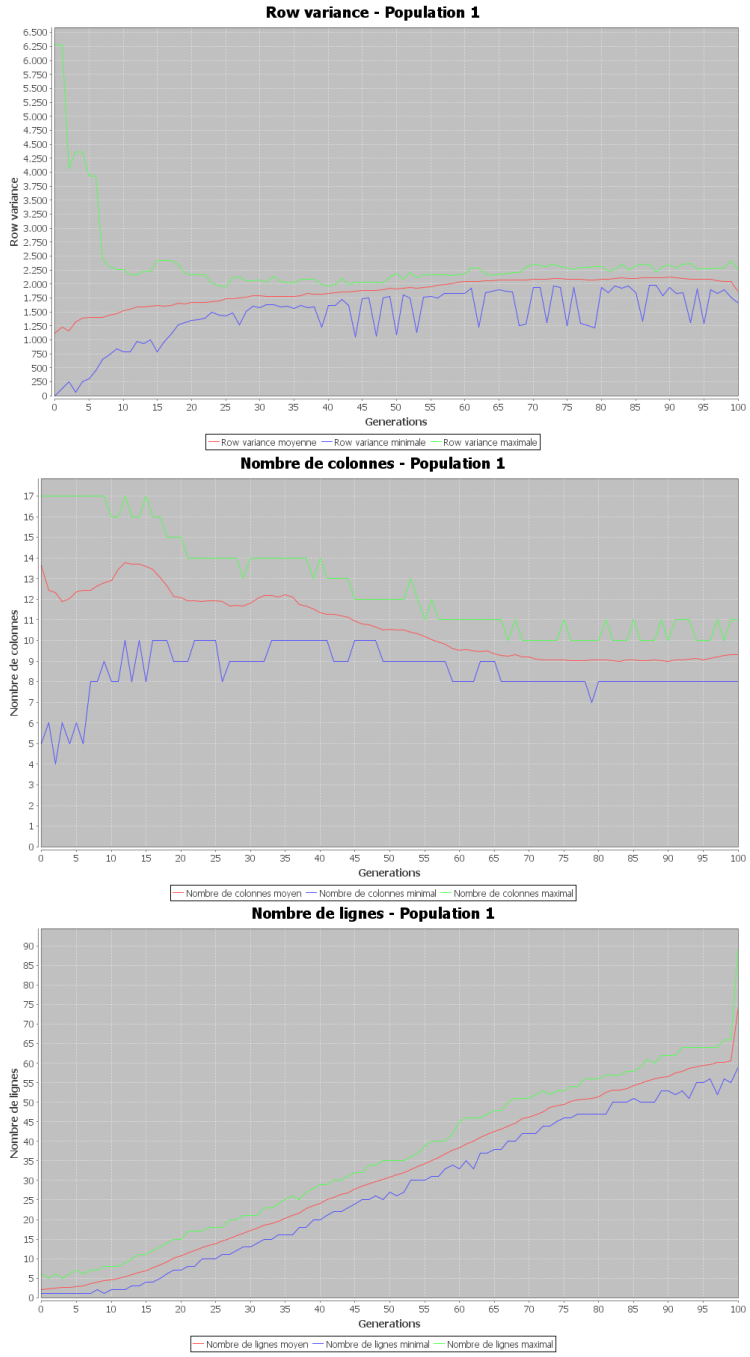


FIGURE 4.4 – Evolution de l'overlapping pour l'ensemble des résultats obtenus pour le Yeast dataset.

Les figures suivantes illustrent quant à elles les évolutions respectives du mean squared residue, du volume, de la row variance, du nombre de colonnes et du nombre de lignes des biclusters lors de l'évolution de la population aboutissant au premier résultat fourni par l'algorithme pour ce dataset [4].



4. En ne considérant uniquement que l'évolution de la population aboutissant au premier bicluster obtenu, on se protège de l'effet des pénalités, causées par les chevauchements etc, et de leur influence dans l'interprétation qui pourrait être faite par la suite sur l'évolution du processus de recherche ainsi que sur l'évolution des caractéristiques envisagées. Cette remarque reste aussi valable pour les analyses de ces évolutions présentées pour le Human Lymphoma dataset et le Colon Cancer dataset.



Ces résultats, illustrant une évolution positive de ces différentes caractéristiques, prouvent que la nouvelle approche utilisée ici réussit à promouvoir des biclusters présentant ces caractères recherchés (mean squared residue, volume, row variance, nombre de colonnes et nombre de lignes). De plus, on observe clairement l'effet de la recherche locale sur le resultat obtenu en fin de cycle.

4.5 Human Lymphoma dataset

4.5.1 Résultats SMOB classique

Voici les résultats obtenus pour une exécution du SMOB dans sa version originale sur le Human Lymphoma dataset. Les détails de ceux-ci se trouvent en annexe C.2.

	SMOB
Mean squared residue	1144,93 (76,72)
Volume	561,88 (268,00)
Row variance	4834,06 (1779,44)
Coverage	10,48%
Overlapping	25,82%
Genes coverage	18,30%
Conditions coverage	100,00%
Execution time (sec)	896,00

4.5.2 Résultats SMOB avec améliorations

Voici les résultats obtenus en combinant le SMOB classique avec les améliorations pour le Human Lymphoma dataset. Les détails de ceux-ci se trouvent en annexe D.2.

	SMOB avec améliorations
Mean squared residue	905,78 (184,11)
Volume	1713,82 (1582,01)
Row variance	6060,62 (6759,59)
Coverage	26,76%
Overlapping	33,42%
Genes coverage	54,54%
Conditions coverage	100,00%
Execution time (sec)	3110,00

4.5.3 Analyse

Cette fois encore, on peut conclure que cette approche fonctionne correctement et répond parfaitement aux attentes en améliorant globalement le SMOB classique. On peut notamment le voir en analysant la figure 4.5 qui illustre justement les résultats obtenus par ces améliorations par rapport à la version classique de l'algorithme. En effet, quelle que soit la caractéristique prise en considération, on obtient encore de meilleures performances par rapport au SMOB classique (à l'exception du temps de calcul et de l'overlapping qui augmentent néanmoins). Comme précédemment, l'augmentation du temps est causée par les améliorations apportées qui nécessitent la réalisation d'un grand nombre de calculs supplémentaires. Et concernant l'augmentation de l'overlapping, on peut aussi dire que celle-ci reste raisonnable compte tenu de la forte augmentation du volume des biclusters résultats et du pourcentage de gènes coverage.

Un ensemble de résultats (provenant des résultats cités en annexe D.2) particulièrement intéressants pour leurs caractéristiques et qui a été trouvé par la nouvelle version du SMOB pour le Human Lymphoma dataset se trouve en annexe F.2.

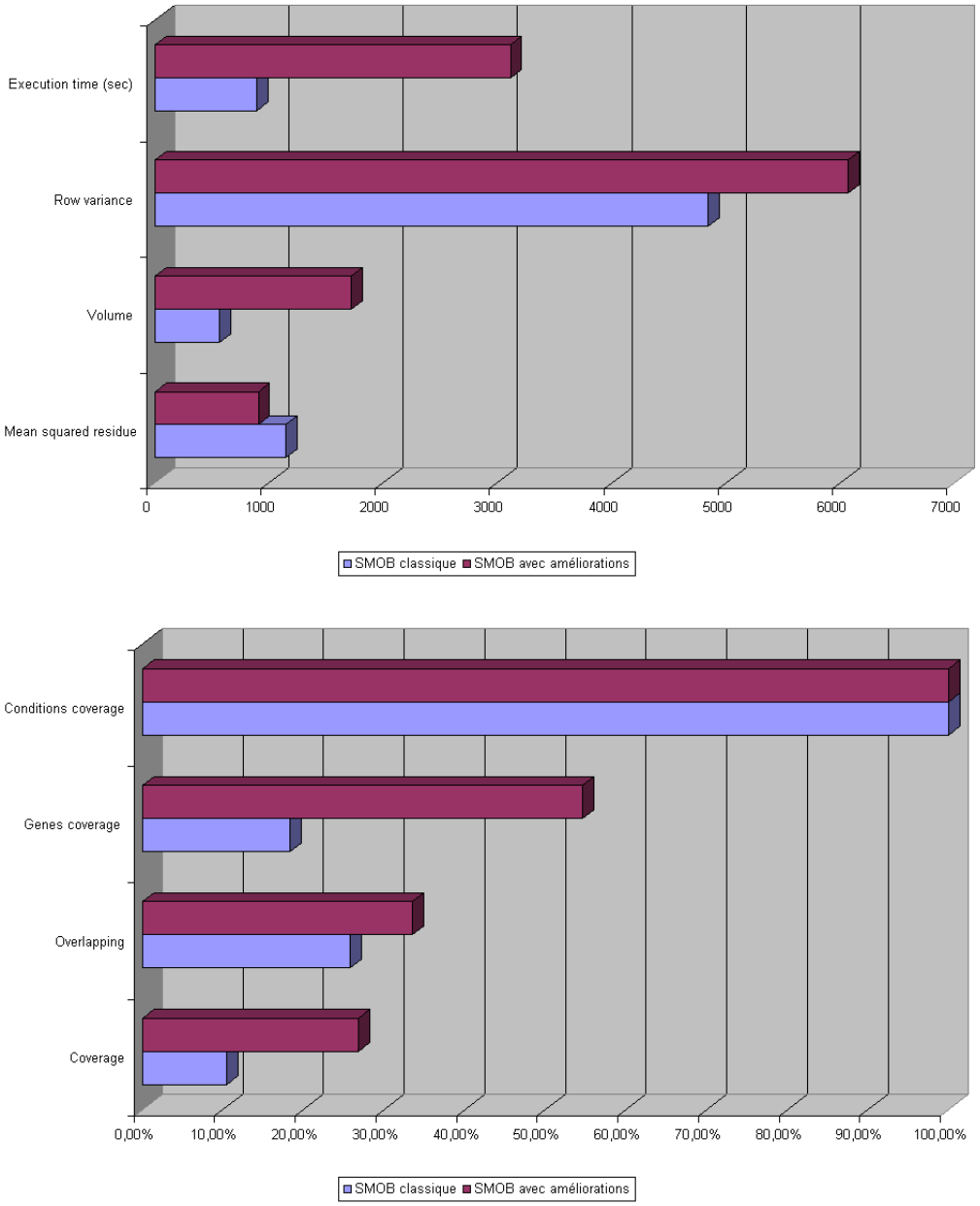


FIGURE 4.5 – Comparaison : SMOB classique avec toutes les améliorations apportées par rapport au SMOB classique pour le Human Lymphoma dataset.

La figure 4.6 illustrant la répartition des résultats obtenus par rapport aux deux caractéristiques des biclusters à considérer en priorité (le mean squared residue et le volume) pour le SMOB classique et le SMOB classique avec toutes les améliorations apportées prouve elle aussi dans ce dataset l'effet bénéfique des améliorations apportées. En effet, on observe que l'ensemble des résultats obtenus par la nouvelle version de l'algorithme se répartit d'une très bonne manière en ayant certains résultats avec de plus grands volumes et globalement de plus petits mean squared residue.

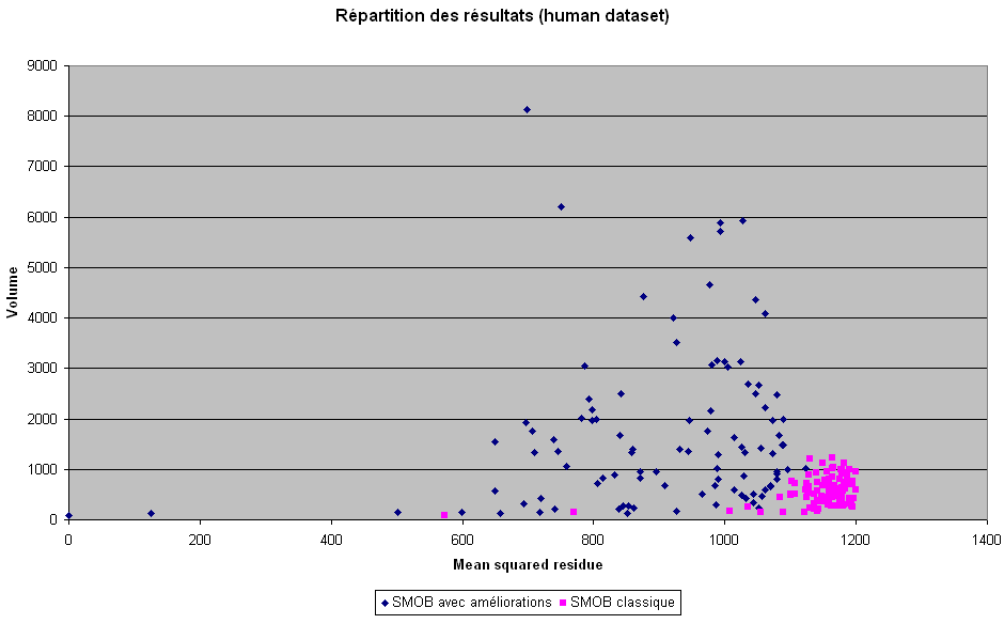


FIGURE 4.6 – Répartition des biclusters résultats obtenus par rapport aux deux caractéristiques à considérer en priorité (le mean squared residue et le volume) pour le SMOB classique et le SMOB classique avec toutes les améliorations apportées pour le Human Lymphoma dataset.

En observant les figures 4.7 et 4.8, on se rend aussi compte des bénéfices que les améliorations apportent concernant l'évolution du coverage et de l'overlapping si l'on prend en compte l'ensemble des résultats obtenus et la progression de leurs apports respectifs à ces caractéristiques. Comme constaté dans les résultats précédents, on remarque que la nouvelle approche fournit un meilleur coverage global, même si l'overlapping devient légèrement plus important qu'auparavant.

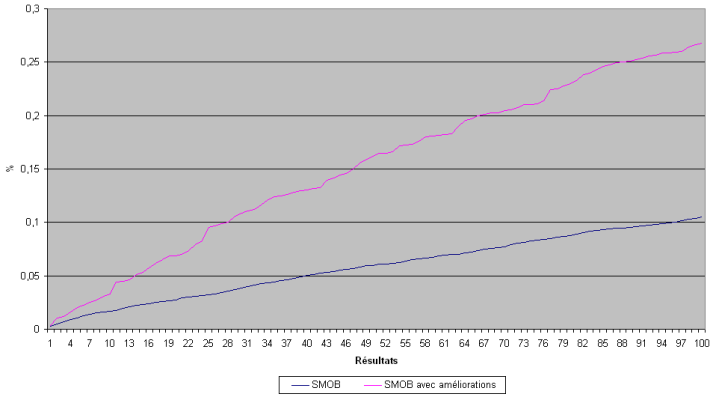


FIGURE 4.7 – Evolution du coverage pour l'ensemble des résultats obtenus avec le Human Lymphoma dataset.

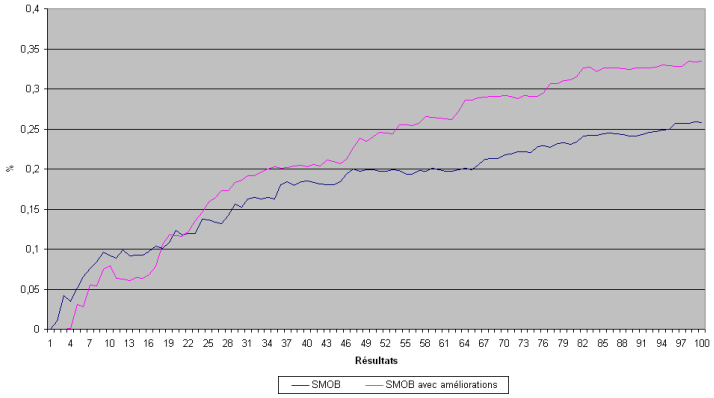
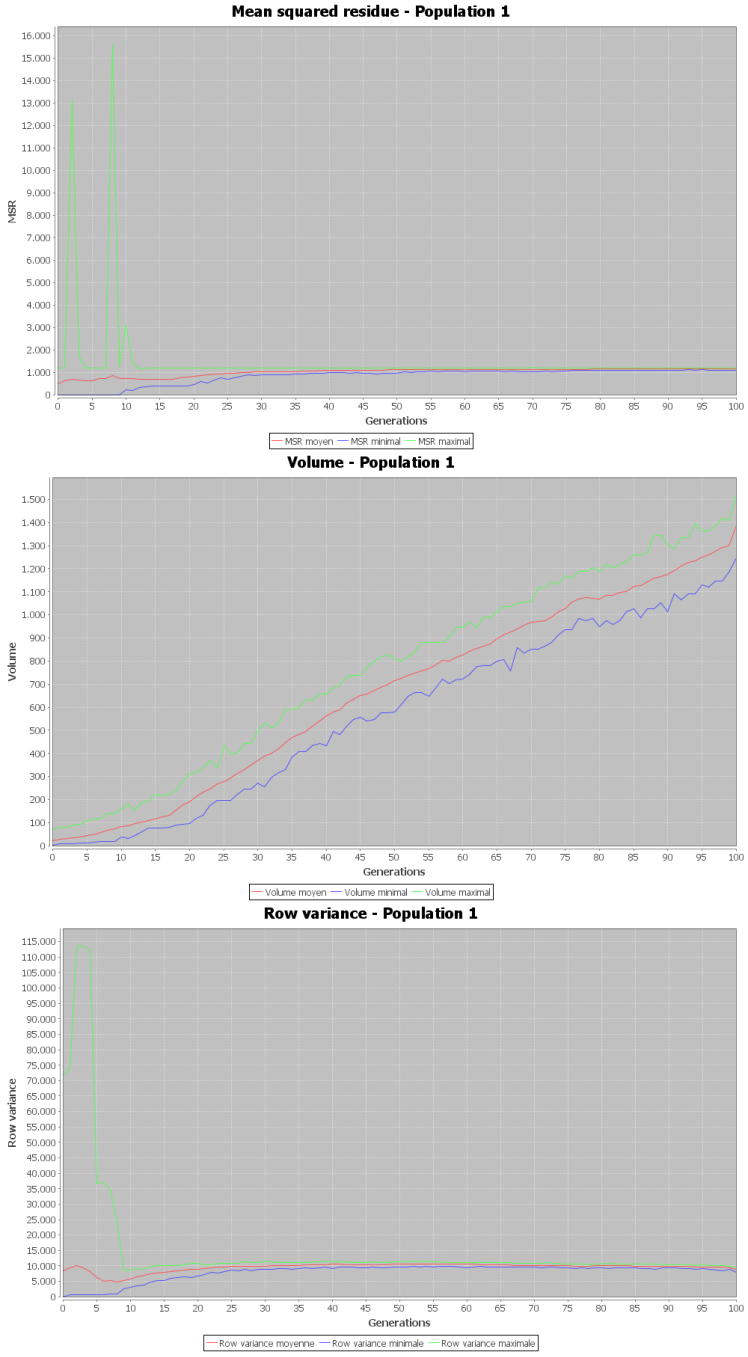


FIGURE 4.8 – Evolution de l'overlapping pour l'ensemble des résultats obtenus avec le Human Lymphoma dataset.

Les figures suivantes illustrent quant à elles les évolutions respectives du mean squared residue, du volume, de la row variance, du nombre de colonnes et du nombre de lignes des biclusters lors de l'évolution de la population aboutissant au premier résultat fourni par l'algorithme pour ce dataset.



Ces résultats, illustrant une évolution positive de ces différentes caractéristiques, prouvent que la nouvelle approche utilisée ici réussit à promouvoir des biclusters présentant ces caractères recherchés (mean squared residue, volume, row variance, nombre de colonnes et nombre de lignes). De plus, on observe clairement l'effet de la recherche locale sur le résultat obtenu en fin de cycle.

4.6 Colon Cancer dataset

4.6.1 Résultats SMOB classique

Voici les résultats obtenus pour une exécution du SMOB dans sa version originale sur le Colon Cancer dataset. Les détails de ceux-ci se trouvent en annexe C.3.

	SMOB
Mean squared residue	484,16 (19,75)
Volume	842,23 (374,76)
Row variance	5218,18 (658,62)
Coverage	32,69%
Overlapping	52,16%
Genes coverage	41,60%
Conditions coverage	100,00%
Execution time (sec)	837,00

4.6.2 Résultats SMOB avec améliorations

Voici les résultats obtenus en combinant le SMOB classique avec les améliorations pour le Colon Cancer dataset. Les détails de ceux-ci se trouvent en annexe D.3.

	SMOB avec améliorations
Mean squared residue	355,98 (69,54)
Volume	672,53 (389,54)
Row variance	6659,39 (2240,99)
Coverage	31,57%
Overlapping	37,26%
Genes coverage	62,50%
Conditions coverage	100,00%
Execution time (sec)	1747,00

4.6.3 Analyse

Les résultats obtenus pour ce dernier dataset par notre nouvelle approche sont légèrement différents de ceux obtenus en utilisant les deux premiers jeux de tests. Cependant, ils confirment toujours le fonctionnement correct de notre approche et répondent donc parfaitement aux attentes en améliorant globalement les résultats obtenus par le SMOB classique. On peut notamment le voir en analysant la figure 4.9 qui illustre justement les résultats obtenus par ces améliorations par rapport à la version classique de l'algorithme. En effet, on note cette fois encore des améliorations aux niveaux du mean squared residue, de la row variance et du gènes coverage, mais aussi, à la différence d'auparavant, au niveau de l'overlapping. Par contre, on va constater ici une diminution du volume moyen des biclusters trouvés ainsi que le coverage total qui va rester plus ou moins constant. Comme précédemment, le temps de calcul total augmente toujours et le conditions coverage reste stable. La diminution du pourcentage d'overlapping s'explique par le fait de l'augmentation combinée du gènes coverage et de la diminution des volumes moyens des biclusters alors que le coverage total reste stable. De ce fait, les biclusters vont se répartir plus uniformément sur la matrice expression et diminuer le nombre de chevauchements. L'augmentation du temps est toujours attribuable aux améliorations apportées qui nécessitent la réalisation d'un grand nombre de calculs supplémentaires.

Un ensemble de résultats (provenant des résultats cités en annexe D.3) particulièrement intéressants pour leurs caractéristiques et qui a été trouvé par la nouvelle version du SMOB pour le Colon Cancer dataset se trouve en annexe F.3.

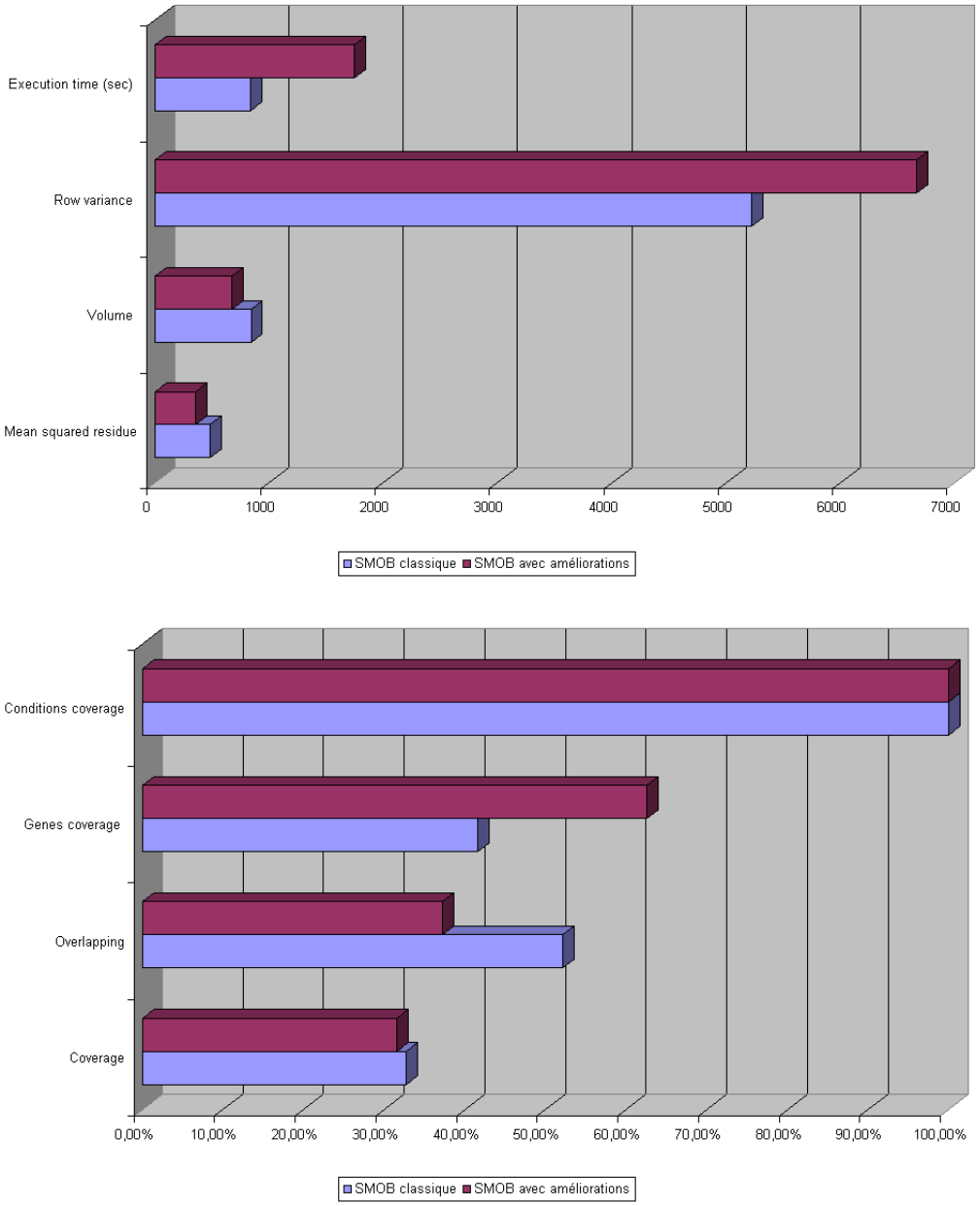


FIGURE 4.9 – Comparaison : SMOB classique avec toutes les améliorations apportées par rapport au SMOB classique pour le Colon Cancer dataset.

Cette fois-ci encore, la figure 4.10 illustrant la répartition des résultats obtenus par rapport aux deux caractéristiques des biclusters à considérer en priorité (le mean squared residue et le volume) pour le SMOB classique et le SMOB classique avec toutes les améliorations apportées prouve globalement elle aussi l'effet bénéfique des améliorations, même si certaines précisions supplémentaires sont à apporter. En effet, on observe que l'ensemble des résultats obtenus par la nouvelle version de l'algorithme se répartit d'une très bonne manière si l'on considère uniquement le mean squared residue, ils ont globalement des valeurs plus basses. Par contre, on ne peut pas vraiment parler d'une réelle amélioration en ce qui concerne les volumes car ceux-ci restent plus ou moins au même niveau que les résultats provenant du SMOB classique.

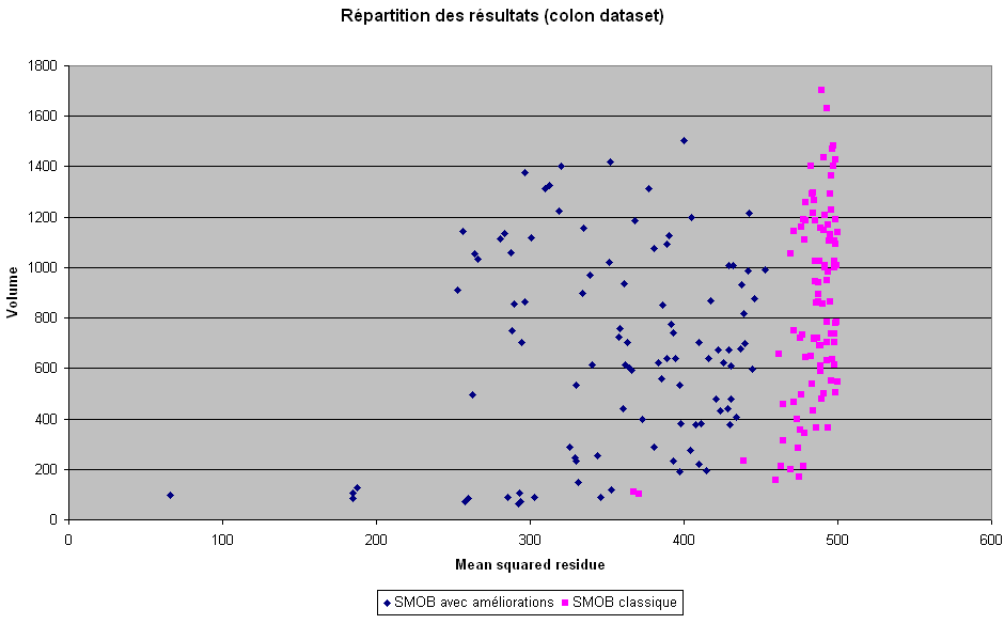


FIGURE 4.10 – Répartition des biclusters résultats obtenus par rapport aux deux caractéristiques à considérer en priorité (le mean squared residue et le volume) pour le SMOB classique et le SMOB classique avec toutes les améliorations apportées pour le Colon Cancer dataset.

En observant les figures 4.11 et 4.12, on se rend aussi compte des effets que les améliorations apportent sur l'évolution du coverage et de l'overlapping si l'on prend en compte l'ensemble des résultats obtenus et la progression de leurs apports respectifs à ces caractéristiques. Comme constaté dans les résultats précédents, on remarque que la nouvelle approche fournit cette fois, à la différence des analyses faites précédemment pour les deux autres datasets, un coverage global plus ou moins stable par rapport à l'approche classique alors que le niveau d'overlapping va devenir moins important que celui obtenu par l'approche classique, ce qui est positif.

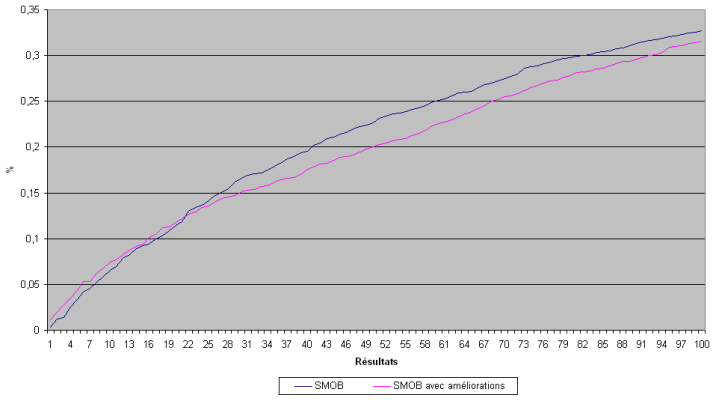


FIGURE 4.11 – Evolution du coverage pour l'ensemble des résultats obtenus avec le Colon Cancer dataset.

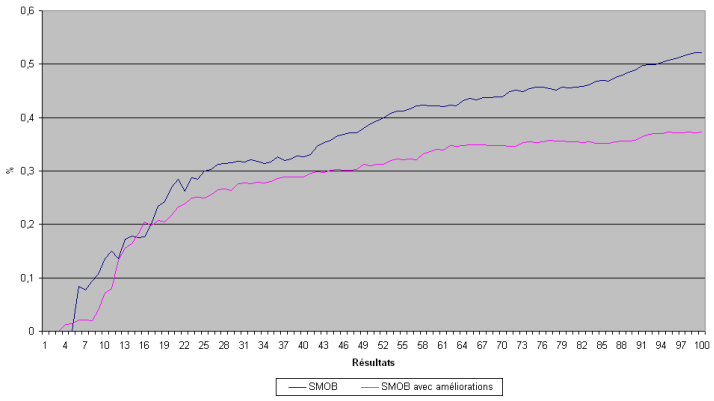
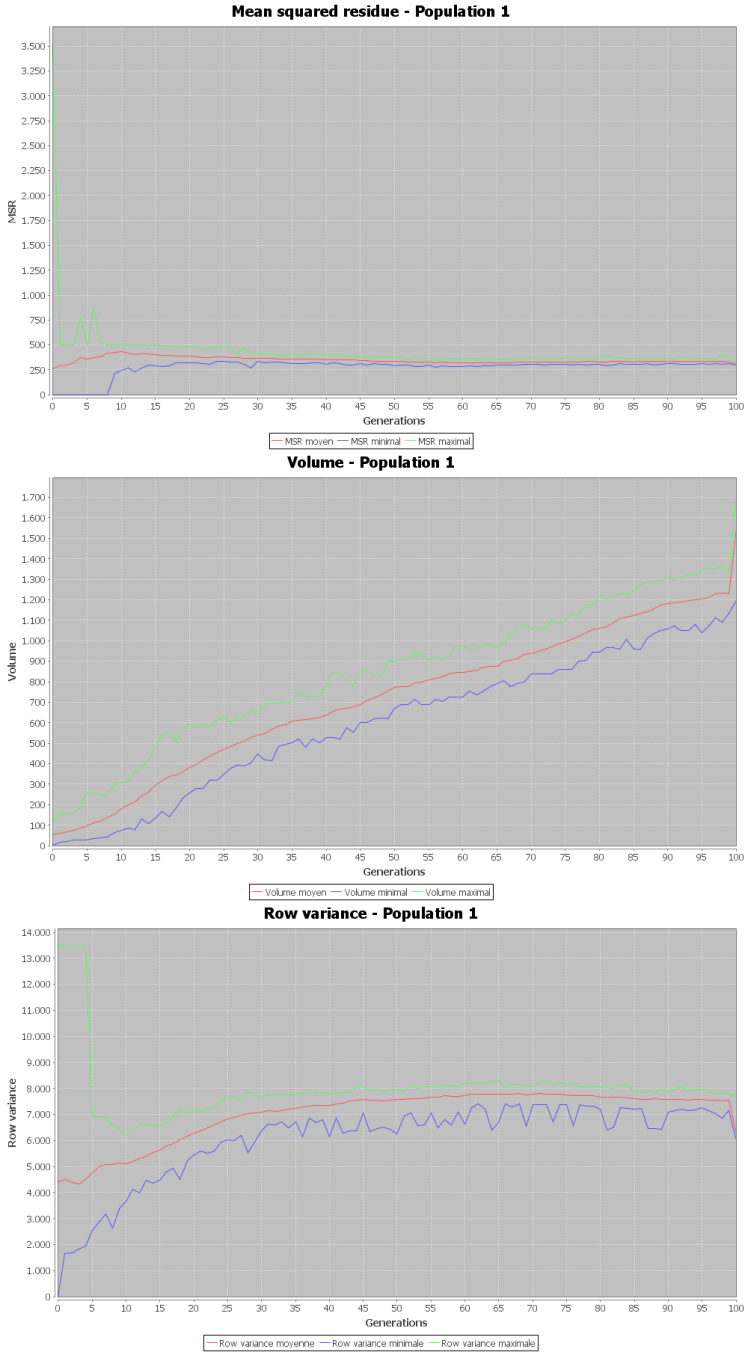
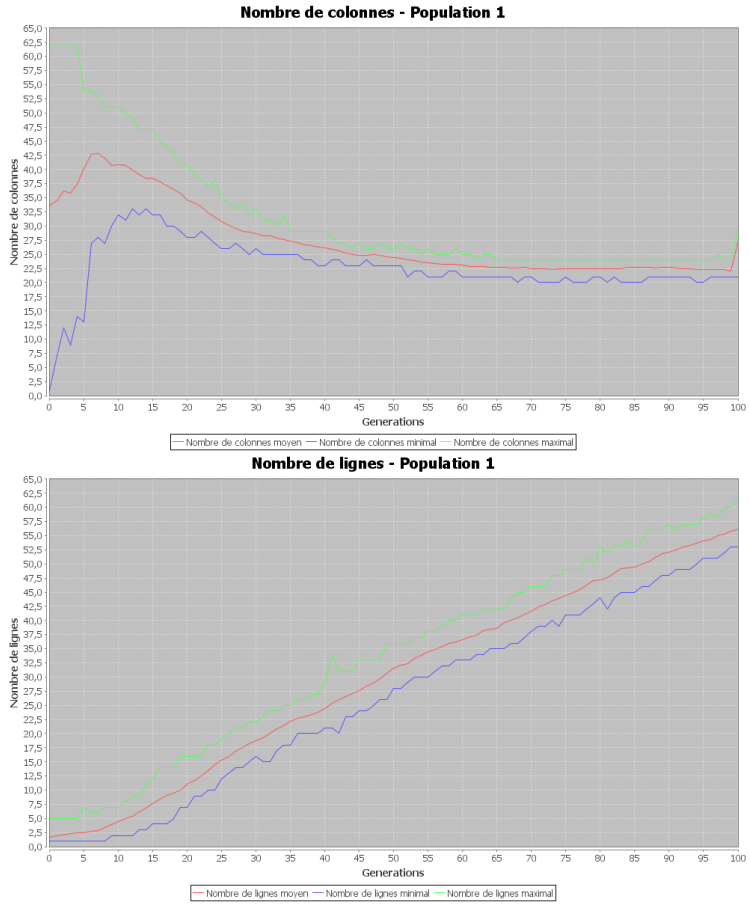


FIGURE 4.12 – Evolution de l'overlapping pour l'ensemble des résultats obtenus avec le Colon Cancer dataset.

Les figures suivantes illustrent quant à elles les évolutions respectives du mean squared residue, du volume, de la row variance, du nombre de colonnes et du nombre de lignes des biclusters lors de l'évolution de la population aboutissant au premier résultat fourni par l'algorithme pour ce dataset.





Ces résultats, illustrant une évolution positive de ces différentes caractéristiques, prouvent que la nouvelle approche utilisée ici réussit à promouvoir des biclusters présentant ces caractères recherchés (mean squared residue, volume, row variance, nombre de colonnes et nombre de lignes). De plus, on observe clairement l'effet de la recherche locale sur le resultat obtenu en fin de cycle.

4.7 Quelques comparaisons

- Voici maintenant une comparaison, illustrée par la figure 4.13, de l'effet des différentes améliorations apportées sur les caractéristiques des résultats obtenus via les trois datasets considérés.

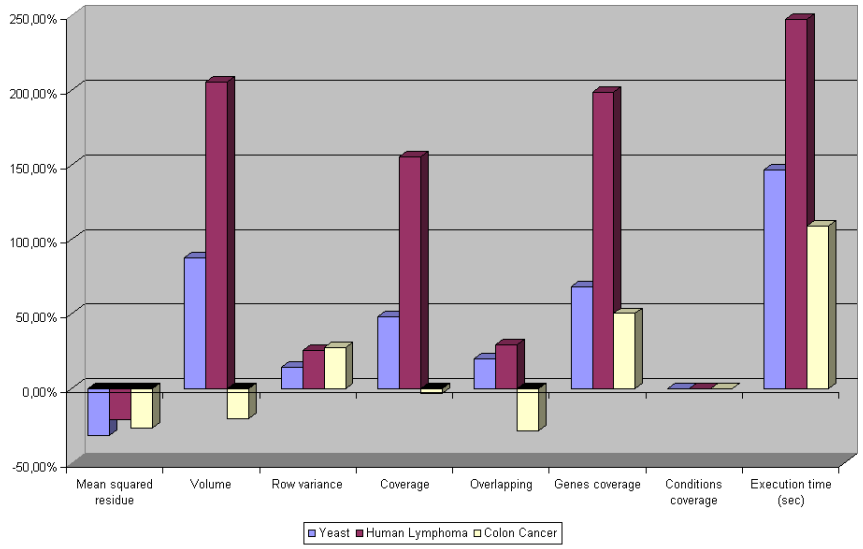


FIGURE 4.13 – Comparaison de l'effet des différentes améliorations apportées sur les caractéristiques des résultats obtenus via les trois datasets considérés.

On observe globalement que les effets bénéfiques se font le plus ressentir sur les résultats obtenus en traitant le Human Lymphoma dataset. En effet, on obtient les augmentations positives les plus fortes pour le volume, le coverage et le gènes coverage mais aussi les augmentations négatives les plus fortes pour l'overlapping et le temps de calcul, de même pour le mean squared residue avec lequel on obtient les moins bonnes performances parmi les trois datasets. Ensuite, concernant le Yeast dataset, on obtient la meilleure amélioration des trois jeux de tests considérés pour la caractéristique fort importante du mean squared residue, tout en conservant de relativement bonnes performances d'amélioration pour les autres caractéristiques, à l'exception de la row variance où on obtient les moins bonnes performances des trois datasets. Finalement, le Colon Cancer dataset présente comparativement des performances globalement moins bonnes que celles des deux autres datasets, surtout pour le volume, le coverage et le gènes coverage où il présente les moins bonnes performances. Par contre, il présente les meilleures performances concernant l'augmentation de la row variance, et au contraire des autres une forte diminution de l'overlapping ainsi que le temps de calcul le moins important des trois datasets.

- Comparons maintenant les résultats obtenus avec notre nouvelle approche par rapport à ceux disponibles pour la méthode de Cheng & Church [2], c'est-à-dire pour le Yeast dataset et pour le Human Lymphoma dataset.

  On obtient pour le Yeast dataset :

	SMOB avec améliorations	Cheng & Church
Mean squared residue	196,69 (50,50)	204,29 (42,78)
Volume	802,81 (781,83)	1576,98 (2178,46)
Row variance	1412,03 (1029,85)	/
Coverage	69,27%	81,47%
Overlapping	58,32%	/
Genes coverage	90,08%	97,12%
Conditions coverage	100,00%	100,00%
Execution time (sec)	1633,00	/

  On obtient pour le Human Lymphoma dataset :

	SMOB avec améliorations	Cheng & Church
Mean squared residue	905,78 (184,11)	850,04 (153,91)
Volume	1713,82 (1582,01)	4595,98 (3353,72)
Row variance	6060,62 (6759,59)	/
Coverage	26,76%	36,81%
Overlapping	33,42%	/
Genes coverage	54,54%	91,58%
Conditions coverage	100,00%	100,00%
Execution time (sec)	3110,00	/

  Comme les valeurs de plusieurs caractéristiques ne sont pas connues pour la méthode de Cheng & Church, et en particulier le niveau d'overlapping, on pourrait croire de premier abord que cette méthode fournit de meilleurs résultats au vu des autres critères. Néanmoins, il faudrait revoir à la baisse notre estimation de la qualité de ces résultats si on suppose un niveau d'overlapping élevé, ce qui reste fort probable[5]. En effet, les plus gros volumes, coverage et gènes coverage obtenus en moyenne sont peut être dus à un fort niveau d'overlapping entre les biclusters résultats. Ou sinon, on observe de bons résultats globaux concernant le mean squared residue pour les deux jeux de tests.

5. La vérification de cette hypothèse, consistant à comparer plus précisément nos résultats avec ceux obtenus par Cheng & Church, serait intéressante à réaliser dans le cadre d'un futur travail.

Chapitre 5

Conclusion

Comme nous venons de le voir dans le dernier chapitre, la nouvelle approche que nous avons mise en place par ce travail, partant de l'algorithme génétique multi-objectif SMOB [5], développé par les Professeurs Federico Divina et Jesus S. Aguilar-Ruiz, à consisté à améliorer l'efficacité de celui-ci en y incorporant des procédures de recherche locale ainsi qu'en utilisant des informations connues et spécifiques au problème au sein des opérateurs des méthodes évolutionnaires. Plus précisément on y a ajouté des procédures de recherche locale basées sur les algorithmes de Cheng et Church [2] ainsi que des opérateurs de mutation intelligents. Ces modifications ont permis d'atteindre une amélioration globale des différentes caractéristiques observées sur les résultats, c'est-à-dire qu'elles améliorent principalement ceux-ci aux niveaux du mean squared residue (en le diminuant), de la row variance (en l'augmentant), du volume (en l'augmentant), des chevauchements (en les diminuant), de la couverture totale (en l'augmentant), des lignes et des colonnes (en les augmentant) en moyenne pour l'ensemble de tous les biclusters trouvés comme solutions pour les différents jeux de tests disponibles (Yeast, Human Lymphoma et Colon Cancer dataset). En effet, il faut se rappeler que nous cherchons des biclusters de taille maximum, avec un mean squared residue plus bas que le δ donné, avec une row variance relativement haute, et avec un faible niveau de chevauchement entre eux. L'analyse plus précise des résultats obtenus faite dans le dernier chapitre, permet d'apporter certaines nuances supplémentaires sur les caractéristiques améliorées ou non par notre approche selon le jeu de données traité, même si l'on peut dire que l'ensemble reste satisfaisant. Nous pouvons cependant quand même préciser certaines limites concernant l'algorithme développé et ses possibilités de pistes pour des travaux futurs. Tout d'abord, il est sans doute possible d'améliorer celui-ci au niveau de ses performances mesurées au niveau du temps de calcul. Certaines optimisations, dans les calculs effectués et dans l'implémentation proposée sont certainement possibles et certains petits problèmes déjà présents dans la version classique de l'algorithme subsistent encore [1]. De plus, comme

1. Les détails de ceux-ci se trouvent en annexe B.

nous l'avons constaté pour les trois jeux de tests envisagés et en particulier pour le Yeast dataset et le Human Lymphoma dataset, il est encore possible d'améliorer l'algorithme afin qu'il diminue au maximum les niveaux d'overlapping pour les résultats obtenus. Comme nous l'avons aussi expliqué auparavant, l'algorithme présenté ici s'inspire des méthodes multi-objectifs mais reste single objectif. Dès lors, un test possible serait de le rendre complètement multi-objectif et d'observer les résultats obtenus de cette manière. Il faudrait donc redéfinir sa fonction de fitness, en sortant de celle-ci certains éléments comme les mesures de distance utilisées que l'on pourrait introduire à d'autres endroits comme par exemple dans les tournois de sélection. Finalement, on peut aussi conseiller des pistes d'exploration supplémentaires permettant d'encore améliorer les résultats de l'algorithme. Ainsi, on pourrait, de la même manière qu'on l'a fait pour les opérateurs de mutation intelligents, modifier les opérateurs de crossover afin de les rendre eux aussi plus performants. Pour cela, on pourrait sélectionner les meilleures parties des deux individus que l'on doit croiser pour qu'ils se les échangent dans le processus. L'initialisation des individus pourrait aussi être améliorée en partant de résultats connus comme étant de bonne qualité ou obtenus par des méthodes de recherche locale exécutées préalablement plutôt que l'initialisation aléatoire utilisée actuellement. Enfin, on pourrait aussi réaliser des tests supplémentaires sur d'autres jeux de données disponibles pour valider encore plus les résultats obtenus avec notre approche. On pourrait également comparer nos résultats avec ceux obtenus par les autres outils existants de l'état de l'art.

Annexe A

Tableau récapitulatif des algorithmes et de leur classification

	Type	Structure	Discovery	Approach
Block Clustering	Constant	(f)	One Set at a Time	Div-and-Conq
δ-biclusters	Coherent Values	(i)	One at a Time	Greedy
FLOC	Coherent Values	(i)	Simultaneous	Greedy
pClusters	Coherent Values	(g)	Simultaneous	Exh-Enum
Plaid Models	Coherent Values	(i)	One at a Time	Dist-Ident
PRMs	Coherent Values	(i)	Simultaneous	Dist-Ident
CTWC	Constant Columns	(i)	One Set at a Time	Clust-Comb
ITWC	Coherent Values	(d),(e)	One Set at a Time	Clust-Comb
DCC	Constant	(b),(c)	Simultaneous	Clust-Comb
δ-Patterns	Constant Rows	(i)	Simultaneous	Greedy
Spectral	Coherent Values	(c)	Simultaneous	Greedy
Gibbs	Constant Columns	(d),(e)	One at a Time	Dist-Ident
OPSMs	Coherent Evolution	(a),(i)	One at a Time	Greedy
SAMBA	Coherent Evolution	(i)	Simultaneous	Exh-Enum
xMOTIFs	Coherent Evolution	(a),(i)	Simultaneous	Greedy
OP-Clusters	Coherent Evolution	(i)	Simultaneous	Exh-Enum

[1]

Légende pour les structures :

(a) Single Bicluster
(b) Exclusive row and column biclusters
(c) Non-Overlapping biclusters with Checkerboard structure
(d) Exclusive-rows biclusters
(e) Exclusive-columns biclusters
(f) Non-Overlapping biclusters with tree structure
(g) Non-Overlapping non-exclusive biclusters
(h) Overlapping biclusters with hierarchical structure
(i) Arbitrarily positioned overlapping biclusters

Annexe B

Commentaire général sur les résultats obtenus

Afin que notre analyse des résultats obtenus soit la plus complète possible, il nous faut encore aborder un problème qui pouvait d'ailleurs survenir avec l'implémentation classique du SMOB mais qui subsiste encore dans la nouvelle version développée dans le cadre de notre travail. Signalons toutefois que la survenance de ce problème est relativement rare, on ne le rencontre que deux fois sur l'ensemble des résultats présentés par la suite. Ce problème consiste donc plus précisément en la possibilité d'obtenir comme résultat des biclusters ayant une seule ligne ou une seule colonne.

En étudiant plus précisément l'algorithme, on se rend compte, au niveau du SMOB classique, que ce problème ne peut survenir qu'à cause des différents opérateurs de crossover et de celui du standard mutation operator, qui sont les seuls à pouvoir diminuer le nombre de lignes ou de colonnes contenues dans un bicluster. Une solution possible à ce problème serait de pénaliser dans la fonction de fitness les individus composés d'une ligne ou d'une colonne. Néanmoins, cette solution va aussi avoir un fort effet sur la diversité de la population qui diminuera alors fortement. En effet, cette ligne ou colonne aurait pu servir à enrichir par la suite les individus de la population au cours des générations suivantes.

D'autre part, cette situation est d'autant plus rare que l'individu incriminé doit subsister jusqu'à la dernière génération et qu'une des autres caractéristiques que l'on cherche à optimiser simultanément est le volume. Donc, n'avoir qu'une ligne ou colonne va à l'inverse de cela. Une fois les améliorations incorporées à l'algorithme, la situation va encore être un peu accentuée, car les possibilités de suppression de lignes ou de colonnes vont être renforcées par l'utilisation des méthodes de recherche locale.

Finalement, une autre remarque générale fort importante par rapport à ce problème est qu'une fois qu'un individu n'est plus constitué que d'une seule ligne ou d'une seule colonne, celui-ci se retrouve dans une impasse. En effet, le mean squared residue de cet individu est alors de 0, ce qui est donc parfait pour l'algorithme et il est donc très difficile, voir impossible, d'encore améliorer l'individu en lui ajoutant des lignes ou colonnes sans en augmenter cette caractéristique.

Annexe C

Résultats SMOB classique

C.1 Yeast dataset

```
result_yeast_11122009112851.txt         Temps total d'execution: 663 sec
Coverage: 0.46754915
Overlapping: 0.48649827
Genes: 0.5367545
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 32.87 | 13.07 | 428.68 | 287.1336 | 1239.3082 | 0.8157019
Ecart type | 15.277865 | 1.7161291 | 202.38313 | 14.812762 | 650.99945 | 0.5367827
1 | 12 | 15 | 180 | 291.16772 | 1909.0558 | 0.076372035
2 | 46 | 14 | 644 | 299.06808 | 1076.7848 | 0.09064024
3 | 32 | 12 | 384 | 299.1226 | 1356.2955 | 0.16006398
4 | 49 | 13 | 637 | 294.95828 | 1212.8907 | 0.2584182
5 | 45 | 15 | 675 | 296.96512 | 927.5956 | 0.30427223
6 | 44 | 12 | 528 | 284.50156 | 1249.1404 | 0.14923458
7 | 32 | 16 | 512 | 296.36215 | 985.219 | 0.12701672
8 | 49 | 13 | 637 | 298.69125 | 992.348 | 0.3866142
9 | 20 | 14 | 280 | 283.02594 | 1605.7263 | 0.13093981
10 | 49 | 14 | 686 | 289.8762 | 912.05383 | 0.43575668
11 | 28 | 12 | 336 | 291.85065 | 1546.8843 | 0.18347095
12 | 16 | 12 | 192 | 283.94205 | 1712.985 | 0.24137877
13 | 34 | 13 | 442 | 299.16342 | 1223.8003 | 0.3488326
14 | 52 | 16 | 832 | 291.22858 | 816.5226 | 0.7873572
15 | 24 | 15 | 360 | 296.46442 | 952.2144 | 0.66340494
16 | 53 | 9 | 477 | 293.1812 | 1402.8099 | 0.6360459
17 | 51 | 12 | 612 | 293.6811 | 1060.0405 | 0.8988714
18 | 29 | 16 | 464 | 293.0753 | 1140.1031 | 0.36150151
19 | 22 | 9 | 198 | 269.6964 | 1955.3949 | 0.5234093
20 | 28 | 14 | 392 | 292.0298 | 1064.8651 | 0.55010295
```

29	21	39	12	468	294.5035	969.5689	0.51740074
30	22	75	12	900	287.2696	926.33624	0.90461195
31	23	51	14	714	295.27277	869.0385	0.88165885
32	24	59	13	767	293.005	840.5384	1.0075519
33	25	26	14	364	282.2762	1141.6356	0.6585972
34	26	23	14	322	295.0306	1331.1385	0.40177318
35	27	41	14	574	289.82632	892.49445	1.0864785
36	28	38	14	532	287.33887	890.9947	0.89935434
37	29	40	12	480	294.49078	959.0623	0.90523064
38	30	28	14	392	296.67493	1219.0886	1.5343783
39	31	15	13	195	296.1736	2525.2078	0.0631758
40	32	35	10	350	270.81546	1139.111	0.62553185
41	33	25	13	325	257.97598	915.268	0.32768667
42	34	19	12	228	296.34467	1063.9459	0.37022167
43	35	47	11	517	287.88068	1042.9904	1.2551011
44	36	35	15	525	297.0777	778.5307	0.55598676
45	37	21	16	336	293.00064	897.7626	0.41911805
46	38	22	13	286	289.90652	1286.443	0.36511472
47	39	55	13	715	297.8993	830.2741	1.1974579
48	40	14	13	182	287.19534	1155.4342	0.6693715
49	41	55	15	825	290.43454	727.9787	1.1494691
50	42	25	14	350	288.78207	1068.7559	0.6587345
51	43	46	11	506	293.33395	887.11847	0.87976974
52	44	36	14	504	298.78198	892.54205	0.5344561
53	45	53	11	583	291.42038	1081.4276	1.3310926
54	46	38	12	456	257.74536	872.6021	0.099798605
55	47	11	10	110	274.2365	1871.0162	0.18982875
56	48	43	15	645	299.979	778.2377	1.3309059
57	49	32	13	416	296.55945	1215.3488	0.8906251
58	50	35	11	385	291.41116	890.33295	0.6005033
59	51	37	14	518	290.59668	797.94946	0.7632909
60	52	44	14	616	290.81235	838.03546	0.4177121
61	53	30	11	330	281.94928	1206.9381	0.72854275
62	54	24	15	360	295.42657	1334.5558	1.4506646
63	55	5	12	60	238.67169	3997.2014	0.052767448
64	56	71	11	781	280.888	855.9551	1.4969761
65	57	14	9	126	283.28998	2388.7888	0.50903124
66	58	28	14	392	298.1372	846.5028	0.7919735
67	59	43	15	645	298.88846	721.0687	1.5498108
68	60	7	14	98	236.10233	3411.4424	0.06966472
69	61	37	13	481	295.44635	827.5338	0.84132975
70	62	14	14	196	279.54904	2256.9082	0.31531692
71	63	23	13	299	280.57617	927.38794	0.90187514
72	64	59	14	826	292.73236	770.96497	1.5165943

73	65	23	12	276	279.92087	987.7432	0.65860236
74	66	31	10	310	291.88855	1128.0714	0.7609978
75	67	45	13	585	284.77344	804.8258	1.473231
76	68	16	14	224	297.49405	1756.8693	0.39562225
77	69	20	14	280	279.82556	966.0191	0.779935
78	70	11	12	132	271.26025	3351.6633	0.29786956
79	71	24	13	312	295.27664	1021.749	1.081681
80	72	30	12	360	287.66446	1008.443	0.70934117
81	73	51	14	714	294.58243	745.09015	1.4491844
82	74	37	11	407	285.3198	1100.3534	1.1536318
83	75	51	11	561	288.11304	867.77954	1.1126313
84	76	34	15	510	299.7847	777.5669	1.4805071
85	77	41	15	615	294.12732	772.4981	1.1659378
86	78	35	14	490	295.03058	835.8252	1.7447127
87	79	31	16	496	284.69537	700.54486	1.4037507
88	80	50	13	650	292.33228	698.2687	1.5142289
89	81	49	13	637	298.1876	938.88086	2.3174996
90	82	18	11	198	285.67926	1783.2715	0.79199976
91	83	64	13	832	299.25418	844.62164	1.9399538
92	84	45	11	495	274.4809	953.2706	1.7049339
93	85	21	13	273	291.08704	1169.637	1.6906395
94	86	3	15	45	219.25241	2143.4993	0.29230157
95	87	24	15	360	297.1998	891.11096	0.73722494
96	88	16	10	160	280.16922	1627.1132	1.1153804
97	89	34	15	510	298.8621	839.1678	1.7206343
98	90	30	12	360	299.2474	1089.1311	1.91604
99	91	16	14	224	280.35403	1227.964	1.4889412
100	92	26	15	390	299.58353	1018.9714	1.4433864
101	93	24	13	312	280.21017	769.1556	0.60275
102	94	12	15	180	297.72784	1542.3075	0.5524342
103	95	18	9	162	286.49948	1440.7856	0.5510245
104	96	2	13	26	230.10945	4121.781	0.04926751
105	97	27	12	324	295.5201	2408.3687	0.62442136
106	98	7	14	98	250.60371	1834.5552	0.14752242
107	99	49	12	588	281.9088	843.91675	2.0676363
108	100	39	16	624	291.56985	675.8066	1.6040773

C.2 Human Lymphoma dataset

result_human-lymphoma_01122009091552.txt Temps total d'execution : 896 sec
Coverage: 0.104860075
Overlapping: 0.25829056
Genes: 0.18306011
Conditions: 1.0

Bicluster	Rows	Columns	Volume	Residue	Row Variance	Fitness
Moyenne	12.85	45.96	561.88	1144.9349	4834.064	0.48326832
Ecart type	7.4288287	6.445029	268.00067	76.7296	1779.4425	0.30868918
1	22	45	990	1179.2906	6106.6436	0.061277166
2	22	42	924	1182.4948	4936.4697	0.11305874
3	22	47	1034	1166.0725	4804.711	0.14455804
4	12	51	612	1170.0128	4702.592	0.19412741
5	27	27	729	1106.9563	4572.7705	0.1944054
6	19	42	798	1175.6332	3869.032	0.19644928
7	18	36	648	1127.4338	4530.1655	0.20910126
8	16	47	752	1102.9025	4539.3247	0.15799828
9	8	59	472	1168.4166	5525.602	0.49037433
10	6	46	276	1179.4974	4727.6025	0.14963426
11	4	53	212	1136.5907	14082.215	0.21046872
12	15	56	840	1163.974	4019.3008	0.34101453
13	11	50	550	1128.0819	3771.022	0.1277769
14	13	47	611	1164.9418	5164.744	0.28311688
15	7	45	315	1175.4501	4106.7	0.108764976
16	9	55	495	1178.7356	5498.6865	0.34835768
17	11	52	572	1180.7157	4545.0938	0.48337623
18	6	46	276	1169.8369	3098.6572	0.30323955
19	10	50	500	1107.5011	7184.1714	0.3761924
20	13	41	533	1126.7302	4351.634	0.4461986
21	14	48	672	1157.724	4371.8364	0.10087327
22	9	50	450	1154.5405	3950.404	0.37543678
23	3	46	138	770.5309	14566.018	0.26161602
24	15	54	810	1158.3267	5344.1616	0.66829544
25	6	59	354	1193.4657	2984.439	0.4173121
26	5	53	265	1165.1825	5646.6665	0.19972397
27	11	53	583	1199.4934	4227.1636	0.29000264
28	16	45	720	1125.694	5035.276	0.5293189
29	23	43	989	1190.5906	5040.954	0.92777836
30	9	47	423	1195.9763	4999.9736	0.19211406
31	36	31	1116	1148.7874	4893.9263	0.5836537
32	12	49	588	1123.2731	5855.805	0.50125045
33	13	53	689	1191.3417	4036.7107	0.34844252
34	9	48	432	1189.4069	7277.721	0.534568

35	6	39	234	1129.2843	5423.3022	0.15794845
36	34	36	1224	1164.6676	5415.2847	0.79855156
37	9	50	450	1125.6108	9495.583	0.77848154
38	12	44	528	1159.9906	3020.7542	0.28984758
39	20	39	780	1155.0977	4213.9297	0.44140118
40	16	47	752	1194.8329	4677.452	0.45475113
41	6	42	252	1035.9536	4101.654	0.17598045
42	14	46	644	1163.203	3842.1152	0.3997525
43	4	42	168	1008.31036	8260.817	0.49762416
44	19	27	513	1101.1113	3931.8655	0.50141263
45	22	42	924	1139.7721	4197.0913	0.6527581
46	12	48	576	1141.3282	5348.9497	1.1064191
47	13	49	637	1163.5238	4717.4517	1.3905904
48	10	47	470	1147.0547	3753.5735	0.112237
49	16	43	688	1150.0417	3435.9712	0.3580202
50	5	50	250	1195.6516	3980.2917	0.4492082
51	9	42	378	1141.0955	4329.926	0.30796546
52	5	44	220	1143.0956	4645.2954	0.53932524
53	7	57	399	1178.7438	3105.3982	0.38717943
54	3	49	147	1121.1624	7362.455	0.107257046
55	17	41	697	1182.1646	3939.2974	0.13221464
56	14	41	574	1159.7515	4134.6274	0.32015613
57	10	51	510	1167.667	4721.351	0.7227336
58	4	41	164	1141.6249	4086.6943	0.25832227
59	18	38	684	1167.3632	4087.484	0.5593206
60	9	54	486	1165.6447	4199.859	0.42714334
61	9	41	369	1158.6703	4817.773	0.06810303
62	3	49	147	1089.4829	6903.53	0.46573317
63	7	52	364	1143.5797	3720.7397	1.0357906
64	13	57	741	1181.8671	3775.9612	0.6265631
65	6	45	270	1193.6469	6569.6147	0.2935914
66	27	35	945	1155.8021	4382.393	0.8808371
67	23	44	1012	1163.2336	3804.3445	0.7425199
68	5	55	275	1162.1643	4091.3909	0.4328866
69	8	49	392	1150.8229	4296.025	0.49632466
70	18	41	738	1141.2908	5105.9097	1.1364442
71	25	36	900	1127.4097	3442.925	0.37176555
72	23	41	943	1199.2573	3667.8298	0.7056331
73	9	54	486	1100.3247	3250.5813	0.37295166
74	10	44	440	1084.6887	4729.798	0.23083493
75	25	45	1125	1181.5421	4017.0579	1.1625298
76	6	49	294	1181.402	4085.524	0.45006022
77	7	50	350	1149.4374	4941.683	0.24096568
78	16	49	784	1152.6322	4310.189	1.0902566

87 79 | 10 | 51 | 510 | 1136.5704 | 5070.382 | 1.0069134
88 80 | 8 | 44 | 352 | 1175.0988 | 3744.6626 | 0.20130187
89 81 | 16 | 40 | 640 | 1178.8201 | 4508.8486 | 0.4166242
90 82 | 38 | 32 | 1216 | 1129.8723 | 4486.9736 | 1.0835891
91 83 | 23 | 43 | 989 | 1177.726 | 3207.0972 | 0.9273418
92 84 | 6 | 51 | 306 | 1157.9047 | 4927.869 | 0.27753338
93 85 | 20 | 47 | 940 | 1187.1746 | 3930.0208 | 1.0123155
94 86 | 8 | 48 | 384 | 1140.3591 | 5430.554 | 0.68800795
95 87 | 3 | 50 | 150 | 1055.5032 | 3906.7542 | 0.27354643
96 88 | 2 | 46 | 92 | 573.4808 | 8014.8315 | 0.1424614
97 89 | 7 | 46 | 322 | 1192.1819 | 4966.7407 | 0.20549725
98 90 | 8 | 45 | 360 | 1151.7853 | 3334.4185 | 0.4969203
99 91 | 6 | 49 | 294 | 1178.1769 | 6821.1235 | 0.9762061
100 92 | 16 | 37 | 592 | 1176.9492 | 4452.0176 | 0.7908195
101 93 | 9 | 46 | 414 | 1154.7811 | 4113.3213 | 0.9181434
102 94 | 6 | 52 | 312 | 1136.4016 | 4918.3716 | 0.65392673
103 95 | 15 | 35 | 525 | 1167.9829 | 3985.4802 | 0.5663182
104 96 | 19 | 48 | 912 | 1178.4564 | 4977.2393 | 1.3096557
105 97 | 20 | 40 | 800 | 1186.2529 | 3133.5994 | 0.5217773
106 98 | 19 | 38 | 722 | 1194.9342 | 3597.408 | 0.5126165
107 99 | 12 | 52 | 624 | 1182.7948 | 3401.8193 | 0.8723309
108 100 | 8 | 55 | 440 | 1163.6852 | 3766.8748 | 0.47271052

C.3 Colon Cancer dataset

```
result_colon_cancer_mod_12122009012522.txt      Temps total d'execution : 837 sec
Coverage: 0.32695967
Overlapping: 0.5216437
Genes: 0.416
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 21.44 | 42.79 | 842.23 | 484.163 | 5218.186 | 1.086063
Ecart type | 11.862816 | 7.4219885 | 374.76587 | 19.7515 | 658.6246 | 0.676967
1 | 7 | 52 | 364 | 493.82773 | 5376.056 | 0.22606742
2 | 34 | 34 | 1156 | 488.78433 | 6221.7534 | 0.051792488
3 | 4 | 53 | 212 | 463.3221 | 5816.0547 | 0.3829059
4 | 35 | 40 | 1400 | 496.9673 | 5168.5737 | 0.064479075
5 | 28 | 36 | 1008 | 499.12354 | 5204.6084 | 0.07162294
6 | 42 | 34 | 1428 | 498.5654 | 5704.7754 | 0.3513258
7 | 11 | 50 | 550 | 495.87198 | 6330.0664 | 0.06818792
8 | 25 | 41 | 1025 | 485.44 | 5092.7944 | 0.35255957
9 | 28 | 41 | 1148 | 490.8109 | 5065.4683 | 0.49416092
10 | 26 | 45 | 1170 | 493.76843 | 4973.0854 | 0.34221303
11 | 17 | 46 | 782 | 492.8139 | 5050.896 | 0.7272717
12 | 26 | 44 | 1144 | 471.68564 | 4552.5254 | 0.15977862
13 | 37 | 30 | 1110 | 478.22095 | 6235.615 | 0.86637807
14 | 24 | 46 | 1104 | 497.94757 | 5448.817 | 0.31112742
15 | 10 | 50 | 500 | 491.08026 | 5669.865 | 0.55731624
16 | 6 | 47 | 282 | 474.53287 | 4910.453 | 0.86147666
17 | 39 | 35 | 1365 | 495.949 | 5356.257 | 0.81634116
18 | 34 | 35 | 1190 | 498.30746 | 5234.183 | 1.1305087
19 | 24 | 42 | 1008 | 491.49548 | 5023.754 | 0.32540843
20 | 38 | 39 | 1482 | 496.96674 | 5537.7437 | 0.78018653
21 | 37 | 35 | 1295 | 484.18475 | 5890.8994 | 1.0170292
22 | 42 | 35 | 1470 | 496.67612 | 4291.385 | 0.11607495
23 | 41 | 35 | 1435 | 490.694 | 5760.1787 | 1.0884682
24 | 7 | 51 | 357 | 475.6107 | 5159.726 | 0.48793438
25 | 34 | 37 | 1258 | 478.99637 | 4865.015 | 0.8390814
26 | 27 | 37 | 999 | 491.7319 | 5141.1025 | 0.42364866
27 | 30 | 41 | 1230 | 495.55365 | 4862.2505 | 1.2134442
28 | 15 | 46 | 690 | 488.6988 | 4864.534 | 0.34989667
29 | 33 | 36 | 1188 | 485.1672 | 4989.3086 | 0.34008798
30 | 16 | 46 | 736 | 498.2107 | 4770.39 | 0.56623363
31 | 15 | 42 | 630 | 493.1935 | 5098.4395 | 0.64031035
32 | 7 | 57 | 399 | 473.61295 | 5098.54 | 0.4696732
33 | 2 | 56 | 112 | 367.04898 | 5390.1084 | 0.27400008
34 | 7 | 52 | 364 | 485.93726 | 5121.9497 | 0.17553368
```

35	45	19	855	490.07962	7418.548	0.58607244
36	27	45	1215	483.87756	4763.0713	1.5822546
37	17	46	782	499.18182	5059.233	0.23650849
38	18	40	720	487.0711	5265.9116	1.6129235
39	27	38	1026	488.39676	5084.069	0.71450746
40	3	56	168	475.18912	6553.733	0.5296589
41	40	35	1400	482.84586	4532.7173	0.7002075
42	39	31	1209	491.43878	5898.838	1.1232189
43	34	31	1054	469.34903	6046.2573	0.67038345
44	15	47	705	493.0181	5105.4844	1.7015212
45	29	39	1131	495.2142	4171.7847	1.2289681
46	13	47	611	489.17987	5634.5166	1.4931531
47	27	38	1026	497.85632	5162.18	0.9726621
48	17	42	714	484.73642	4865.821	1.944027
49	38	34	1292	494.927	5641.0454	2.175066
50	32	37	1184	478.83456	4606.3447	1.3839213
51	29	41	1189	477.4847	4498.7563	0.92588204
52	17	43	731	476.97943	5309.44	0.8277931
53	34	38	1292	483.6366	5476.7676	1.6175644
54	15	41	615	498.158	4952.127	1.9310933
55	7	45	315	464.7244	6123.4507	1.0778075
56	26	33	858	485.8151	5664.967	1.3752749
57	12	53	636	496.63364	5147.75	0.9258607
58	10	48	480	489.33762	5958.1343	0.79903406
59	18	48	864	487.73465	3969.3708	1.0555452
60	13	42	546	499.69025	5737.9907	1.1447027
61	5	47	235	439.1115	6219.8643	0.31576973
62	30	38	1140	499.94183	5357.212	1.7843795
63	22	43	946	485.7529	4211.1377	1.3546681
64	20	39	780	498.28418	5411.8037	1.5699402
65	16	45	720	475.64896	5234.7197	2.3975122
66	15	46	690	488.0142	4590.665	0.7630147
67	40	29	1160	476.57086	4943.226	1.2161678
68	17	38	646	482.38568	5341.458	1.9993225
69	18	41	738	497.62827	4858.3447	1.3420315
70	9	52	468	471.42023	5357.766	0.6073999
71	37	46	1702	489.91516	4381.279	2.0588381
72	25	38	950	492.9007	4937.9385	1.3503473
73	55	23	1265	484.73972	5173.472	0.9030379
74	15	48	720	486.8346	4262.443	1.2959235
75	8	54	432	484.2484	5307.9556	1.3193518
76	14	42	588	488.95157	4542.9595	1.2039187
77	4	50	200	469.4391	5988.5215	0.55437744
78	13	38	494	476.33875	5215.792	0.7894692

87	79	18	48	864	495.1277	4591.7197	1.7250841
88	80	4	53	212	477.4813	6913.4575	0.38572896
89	81	14	46	644	479.29706	4427.177	2.2164917
90	82	3	52	156	459.4644	5138.5996	1.1237447
91	83	16	45	720	484.56937	4231.9233	2.3205361
92	84	16	46	736	496.12134	5226.6626	0.75720173
93	85	14	47	658	461.75778	4269.963	2.641356
94	86	2	51	102	370.63382	7603.3633	0.88734645
95	87	19	47	893	487.36127	4822.0103	1.3549658
96	88	17	44	748	471.31836	4963.386	2.021158
97	89	26	42	1092	498.33047	4691.0845	1.5016662
98	90	37	27	999	498.08942	6147.232	1.4160144
99	91	34	48	1632	493.29578	4330.5225	2.357161
100	92	11	49	539	483.62527	4917.9517	1.9458914
101	93	6	57	342	478.20697	5557.487	1.1846817
102	94	18	39	702	498.13522	5074.8335	1.7428877
103	95	24	46	1104	494.10455	4638.751	2.6087623
104	96	9	51	459	464.4346	5968.7676	1.3544863
105	97	30	37	1110	494.86066	5338.2607	2.6940174
106	98	24	41	984	493.67703	4689.1333	1.8577193
107	99	20	47	940	487.74402	4331.6333	2.4000707
108	100	9	56	504	498.3535	4682.5576	2.0057564

Annexe D

Résultats SMOB et améliorations

D.1 Yeast dataset

```
result_yeast_27112009013023.txt         Temps total d'execution : 1633 sec
Coverage: 0.6927674
Overlapping: 0.5832769
Genes: 0.9008322
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 92.47 | 8.7 | 802.81 | 196.69878 | 1412.0352 | 0.0
Ecart type | 88.34019 | 0.92195404 | 781.8353 | 50.50065 | 1029.8517 | 0.0
1  | 90 | 8 | 720 | 262.42395 | 1714.8605 | 0.0
2  | 71 | 8 | 568 | 141.65248 | 1299.055 | 0.0
3  | 151 | 8 | 1208 | 134.48628 | 880.9609 | 0.0
4  | 118 | 8 | 944 | 121.18376 | 910.9056 | 0.0
5  | 192 | 8 | 1536 | 135.88165 | 405.46738 | 0.0
6  | 73 | 8 | 584 | 132.80489 | 927.85254 | 0.0
7  | 75 | 8 | 600 | 142.04988 | 1563.8344 | 0.0
8  | 38 | 11 | 418 | 255.8587 | 3429.118 | 0.0
9  | 68 | 9 | 612 | 234.61826 | 1395.1578 | 0.0
10  | 135 | 8 | 1080 | 107.3913 | 391.15765 | 0.0
11  | 56 | 12 | 672 | 148.5756 | 882.78625 | 0.0
12  | 90 | 8 | 720 | 186.23439 | 1178.53 | 0.0
13  | 43 | 8 | 344 | 263.38382 | 2850.5784 | 0.0
14  | 91 | 9 | 819 | 219.93066 | 1010.6403 | 0.0
15  | 68 | 8 | 544 | 226.04549 | 1777.7445 | 0.0
16  | 70 | 9 | 630 | 270.04788 | 1706.2762 | 0.0
17  | 115 | 8 | 920 | 179.124 | 638.8051 | 0.0
18  | 52 | 9 | 468 | 220.79074 | 2029.7745 | 0.0
```

27	19	285	8	2280	142.32103	295.85208	0.0
28	20	186	9	1674	193.41043	653.36584	0.0
29	21	26	8	208	225.08562	2345.5752	0.0
30	22	88	9	792	172.58864	948.5948	0.0
31	23	43	11	473	248.30513	1453.5847	0.0
32	24	81	9	729	244.16208	1028.001	0.0
33	25	190	11	2090	244.60349	724.3777	0.0
34	26	50	10	500	267.50885	2880.7734	0.0
35	27	259	10	2590	251.11972	667.03613	0.0
36	28	118	8	944	200.69952	612.0245	0.0
37	29	43	9	387	249.57292	2399.4849	0.0
38	30	27	9	243	268.417	1787.9922	0.0
39	31	46	9	414	239.72017	2004.7832	0.0
40	32	57	9	513	192.69748	1270.6693	0.0
41	33	50	9	450	153.6235	535.38666	0.0
42	34	14	9	126	224.85951	3605.2932	0.0
43	35	64	9	576	198.78407	1048.9028	0.0
44	36	49	11	539	175.03728	802.6424	0.0
45	37	69	9	621	204.96773	1361.9584	0.0
46	38	84	8	672	141.79347	811.14386	0.0
47	39	122	8	976	189.79843	690.396	0.0
48	40	412	10	4120	222.11919	490.54874	0.0
49	41	61	8	488	237.42024	1297.2567	0.0
50	42	127	9	1143	156.0345	756.115	0.0
51	43	389	8	3112	220.1558	302.9747	0.0
52	44	93	9	837	113.81282	355.791	0.0
53	45	45	9	405	271.48288	2129.7627	0.0
54	46	39	9	351	252.79231	1635.0344	0.0
55	47	442	9	3978	222.42523	284.85184	0.0
56	48	45	9	405	255.77197	1942.5496	0.0
57	49	26	8	208	201.32674	1707.7548	0.0
58	50	201	9	1809	210.25905	879.95734	0.0
59	51	58	8	464	222.82092	1575.687	0.0
60	52	56	10	560	245.09822	1466.1871	0.0
61	53	115	8	920	158.73796	773.0633	0.0
62	54	33	10	330	249.69571	1362.5614	0.0
63	55	47	8	376	252.8091	1153.3827	0.0
64	56	7	9	63	171.73038	4827.3555	0.0
65	57	33	8	264	250.50674	1621.2064	0.0
66	58	51	9	459	249.59227	1548.6942	0.0
67	59	47	9	423	265.8375	1605.3676	0.0
68	60	22	8	176	226.88454	1857.2543	0.0
69	61	46	9	414	115.12074	383.16745	0.0
70	62	46	9	414	218.93134	1309.0122	0.0

71	63	56	8	448	202.83713	1146.9691	0.0
72	64	16	8	128	207.82477	3104.2188	0.0
73	65	111	8	888	157.7589	1003.45544	0.0
74	66	34	9	306	230.69592	1305.6097	0.0
75	67	128	9	1152	187.3643	576.1518	0.0
76	68	46	8	368	262.7178	1633.6909	0.0
77	69	54	8	432	147.17703	701.254	0.0
78	70	56	9	504	236.14294	1619.8625	0.0
79	71	353	8	2824	186.38474	253.1327	0.0
80	72	158	9	1422	131.57614	333.1641	0.0
81	73	103	8	824	177.62291	917.29535	0.0
82	74	4	9	36	141.23611	3735.056	0.0
83	75	234	8	1872	137.89226	277.20782	0.0
84	76	86	8	688	224.84169	1449.3868	0.0
85	77	43	9	387	230.51027	1556.614	0.0
86	78	19	8	152	198.47212	2486.524	0.0
87	79	54	8	432	225.55676	888.00696	0.0
88	80	237	9	2133	186.15927	407.1438	0.0
89	81	240	8	1920	148.50056	310.61298	0.0
90	82	50	9	450	188.49841	881.84796	0.0
91	83	90	8	720	213.18071	1415.7231	0.0
92	84	66	8	528	139.89255	719.25287	0.0
93	85	54	8	432	219.41478	1750.3195	0.0
94	86	77	8	616	214.5746	869.1126	0.0
95	87	45	8	360	224.32355	1235.9434	0.0
96	88	12	9	108	224.2392	3240.3027	0.0
97	89	9	8	72	213.13158	3302.0452	0.0
98	90	2	8	16	16.996094	4111.6797	0.0
99	91	256	9	2304	158.3471	292.448	0.0
100	92	65	9	585	215.41975	991.8641	0.0
101	93	127	8	1016	176.87923	481.57886	0.0
102	94	55	11	605	233.96175	833.12244	0.0
103	95	4	9	36	139.5648	3650.8152	0.0
104	96	94	9	846	155.56396	518.19995	0.0
105	97	196	8	1568	165.30441	378.29385	0.0
106	98	20	8	160	213.40149	1375.3961	0.0
107	99	2	11	22	55.103306	5629.927	0.0
108	100	3	6	18	77.88271	1633.4166	0.0

D.2 Human Lymphoma dataset

```
result_human-lymphoma_02122009124238.txt        Temps total d'execution : 3110 sec
Coverage: 0.26761985
Overlapping: 0.33420345
Genes: 0.54545456
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 50.74 | 38.44 | 1713.82 | 905.7804 | 6060.62 | 0
Ecart type | 54.745342 | 12.646202 | 1582.0122 | 184.11203 | 6759.5923 | 0
1 | 32 | 41 | 1312 | 1072.3223 | 8267.966 | 0
2 | 65 | 41 | 2665 | 1051.7012 | 1756.8602 | 0
3 | 56 | 12 | 672 | 908.55664 | 28611.51 | 0
4 | 42 | 40 | 1680 | 1083.3314 | 5023.5166 | 0
5 | 70 | 24 | 1680 | 840.39166 | 4900.19 | 0
6 | 24 | 32 | 768 | 1101.6681 | 4614.805 | 0
7 | 37 | 40 | 1480 | 1089.1265 | 4821.54 | 0
8 | 60 | 16 | 960 | 870.4226 | 20119.385 | 0
9 | 57 | 31 | 1767 | 706.44653 | 4078.2297 | 0
10 | 65 | 11 | 715 | 806.269 | 6054.5825 | 0
11 | 141 | 33 | 4653 | 976.94684 | 1711.8807 | 0
12 | 5 | 31 | 155 | 500.9587 | 19370.008 | 0
13 | 30 | 20 | 600 | 1060.7181 | 10224.765 | 0
14 | 50 | 40 | 2000 | 1089.6238 | 3530.2559 | 0
15 | 22 | 45 | 990 | 1096.1128 | 12118.236 | 0
16 | 42 | 42 | 1764 | 973.8308 | 2908.4993 | 0
17 | 74 | 27 | 1998 | 803.663 | 1737.0111 | 0
18 | 61 | 41 | 2501 | 1047.2811 | 4499.623 | 0
19 | 49 | 41 | 2009 | 781.0356 | 3105.5723 | 0
20 | 6 | 34 | 204 | 837.69226 | 8932.032 | 0
21 | 10 | 47 | 470 | 1055.9178 | 4016.46 | 0
22 | 44 | 45 | 1980 | 1072.6168 | 2805.6743 | 0
23 | 108 | 29 | 3132 | 998.8018 | 2376.1218 | 0
24 | 60 | 37 | 2220 | 1061.2661 | 4131.8335 | 0
25 | 207 | 30 | 6210 | 750.5383 | 903.949 | 0
26 | 44 | 20 | 880 | 831.50684 | 6687.293 | 0
27 | 71 | 27 | 1917 | 697.3643 | 3337.0771 | 0
28 | 25 | 27 | 675 | 1069.2632 | 9700.001 | 0
29 | 87 | 35 | 3045 | 786.8112 | 1137.6055 | 0
30 | 38 | 39 | 1482 | 1086.7687 | 3942.7278 | 0
31 | 45 | 30 | 1350 | 745.8762 | 1571.1815 | 0
32 | 31 | 26 | 806 | 1079.7262 | 4191.36 | 0
33 | 59 | 37 | 2183 | 798.2906 | 1623.1748 | 0
34 | 77 | 35 | 2695 | 1035.8418 | 1843.7633 | 0
```

43	35	29	55	1595	738.8504	1790.1996	0
44	36	11	39	429	1031.7574	5122.477	0
45	37	55	15	825	871.2986	20340.32	0
46	38	45	21	945	894.86163	4431.0303	0
47	39	22	41	902	1079.4717	4128.449	0
48	40	7	59	413	719.32764	14806.593	0
49	41	23	21	483	1025.6276	11228.444	0
50	42	7	38	266	852.9204	5264.388	0
51	43	66	62	4092	1061.3251	1368.5366	0
52	44	24	42	1008	1122.7323	4110.792	0
53	45	22	43	946	1079.5288	3896.0618	0
54	46	64	24	1536	650.0911	3071.2932	0
55	47	143	28	4004	922.20325	2531.9028	0
56	48	134	33	4422	875.4839	1129.5972	0
57	49	30	48	1440	1025.3618	3626.5244	0
58	50	55	36	1980	946.5228	2283.5747	0
59	51	56	35	1960	798.2715	1689.8163	0
60	52	3	41	123	657.6373	9520.284	0
61	53	15	45	675	984.77924	5814.3843	0
62	54	132	33	4356	1046.8878	1344.1903	0
63	55	16	36	576	649.8698	1926.9115	0
64	56	9	36	324	692.8643	18737.674	0
65	57	54	40	2160	978.71967	3465.5503	0
66	58	117	30	3510	925.5438	2117.6719	0
67	59	6	39	234	1052.0454	10764.351	0
68	60	8	36	288	986.9442	11570.025	0
69	61	4	58	232	860.3599	7179.4775	0
70	62	20	25	500	1044.268	6503.8545	0
71	63	143	40	5720	992.40454	1186.0992	0
72	64	180	33	5940	1027.364	1621.2834	0
73	65	13	46	598	1014.4118	10528.218	0
74	66	49	49	2401	793.1224	1921.0365	0
75	67	31	46	1426	1055.405	4639.4336	0
76	68	44	24	1056	759.0219	2002.2876	0
77	69	1	92	92	3.40E-12	37262.062	0
78	70	44	32	1408	930.6481	3184.6094	0
79	71	5	32	160	925.6372	5490.235	0
80	72	13	66	858	1028.2308	6797.708	0
81	73	62	51	3162	988.7404	1571.7532	0
82	74	3	49	147	598.51953	12210.233	0
83	75	3	52	156	717.4056	11337.891	0
84	76	53	57	3021	1003.7863	1411.8083	0
85	77	387	21	8127	698.51715	757.29065	0
86	78	14	37	518	965.07825	16396.672	0

```
87   79 | 64 | 49 | 3136 | 1023.3917 | 2208.0764 | 0
88   80 | 38 | 37 | 1406 | 859.3437 | 1621.9655 | 0
89   81 | 55 | 45 | 2475 | 1079.6174 | 2174.3408 | 0
90   82 | 203 | 29 | 5887 | 992.52045 | 1463.3164 | 0
91   83 | 41 | 33 | 1353 | 943.4268 | 2800.4402 | 0
92   84 | 34 | 48 | 1632 | 1014.16675 | 3988.2207 | 0
93   85 | 96 | 32 | 3072 | 980.15814 | 1405.46 | 0
94   86 | 16 | 51 | 816 | 813.83545 | 1216.7405 | 0
95   87 | 25 | 53 | 1325 | 858.47253 | 1312.2284 | 0
96   88 | 7 | 49 | 343 | 1043.0526 | 6651.8726 | 0
97   89 | 4 | 51 | 204 | 740.6333 | 7467.3 | 0
98   90 | 37 | 35 | 1295 | 989.12946 | 3524.8 | 0
99   91 | 42 | 24 | 1008 | 987.41455 | 2423.0632 | 0
100  92 | 42 | 32 | 1344 | 709.226 | 1054.4261 | 0
101  93 | 20 | 33 | 660 | 1070.3 | 8561.785 | 0
102  94 | 50 | 50 | 2500 | 842.3316 | 1612.2413 | 0
103  95 | 3 | 44 | 132 | 851.53125 | 8120.359 | 0
104  96 | 6 | 47 | 282 | 844.85455 | 11633.593 | 0
105  97 | 2 | 66 | 132 | 124.82367 | 37464.418 | 0
106  98 | 130 | 43 | 5590 | 947.47736 | 1177.3608 | 0
107  99 | 21 | 64 | 1344 | 1030.6771 | 2344.871 | 0
108  100 | 22 | 37 | 814 | 989.1286 | 3097.4653 | 0
```

D.3 Colon Cancer dataset

```
result_colon_cancer_mod_12122009015815.txt      Temps total d'execution : 1747 sec
Coverage: 0.31570968
Overlapping: 0.37261164
Genes: 0.625
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 30.14 | 25.32 | 672.53 | 355.9875 | 6659.392 | 0
Ecart type | 18.615595 | 8.643934 | 389.54926 | 69.54169 | 2240.9988 | 0
1 | 53 | 26 | 1378 | 296.3116 | 6426.382 | 0
2 | 53 | 19 | 1007 | 429.28262 | 8283.081 | 0
3 | 42 | 26 | 1092 | 388.75317 | 6438.7544 | 0
4 | 29 | 34 | 986 | 441.3838 | 4992.251 | 0
5 | 32 | 31 | 992 | 452.5957 | 5849.963 | 0
6 | 51 | 24 | 1224 | 318.56772 | 7001.3438 | 0
7 | 3 | 29 | 87 | 285.09845 | 12258.972 | 0
8 | 44 | 26 | 1144 | 256.02594 | 3749.954 | 0
9 | 43 | 19 | 817 | 439.01273 | 8513.052 | 0
10 | 48 | 25 | 1200 | 405.057 | 6280.9185 | 0
11 | 15 | 27 | 405 | 433.76752 | 5837.034 | 0
12 | 53 | 25 | 1325 | 312.3414 | 5870.969 | 0
13 | 43 | 26 | 1118 | 300.6167 | 4108.9 | 0
14 | 57 | 17 | 969 | 338.65182 | 8058.68 | 0
15 | 25 | 28 | 700 | 439.437 | 7368.373 | 0
16 | 41 | 32 | 1312 | 376.74182 | 5007.411 | 0
17 | 33 | 15 | 495 | 262.2767 | 1242.1293 | 0
18 | 63 | 18 | 1134 | 283.67502 | 6899.651 | 0
19 | 3 | 49 | 147 | 331.50613 | 7000.0723 | 0
20 | 30 | 31 | 930 | 437.63174 | 6020.3564 | 0
21 | 48 | 19 | 912 | 252.52415 | 5562.2827 | 0
22 | 35 | 25 | 875 | 445.49307 | 7634.6733 | 0
23 | 37 | 23 | 851 | 386.3443 | 7803.043 | 0
24 | 26 | 26 | 676 | 436.4647 | 7596.81 | 0
25 | 16 | 16 | 256 | 343.84055 | 6866.1104 | 0
26 | 90 | 10 | 900 | 334.16617 | 9161.97 | 0
27 | 53 | 21 | 1113 | 280.27872 | 6424.6025 | 0
28 | 21 | 18 | 378 | 407.78052 | 7799.0977 | 0
29 | 5 | 47 | 235 | 329.94284 | 5373.7954 | 0
30 | 46 | 23 | 1058 | 287.6805 | 5659.9995 | 0
31 | 17 | 26 | 442 | 360.57028 | 6195.1074 | 0
32 | 4 | 27 | 108 | 292.88162 | 9797.005 | 0
33 | 21 | 29 | 609 | 430.88736 | 5435.415 | 0
34 | 3 | 42 | 126 | 187.51736 | 5664.038 | 0
```

43 35 | 29 | 30 | 870 | 417.05072 | 5591.894 | 0
44 36 | 68 | 15 | 1020 | 351.70224 | 7802.3486 | 0
45 37 | 38 | 14 | 532 | 329.87442 | 7624.868 | 0
46 38 | 2 | 37 | 74 | 293.5592 | 9772.619 | 0
47 39 | 35 | 17 | 595 | 366.10852 | 5116.973 | 0
48 40 | 38 | 32 | 1216 | 442.50354 | 5640.485 | 0
49 41 | 23 | 26 | 598 | 444.40714 | 6405.4365 | 0
50 42 | 54 | 22 | 1188 | 368.15314 | 6232.777 | 0
51 43 | 2 | 43 | 86 | 185.01433 | 12163.94 | 0
52 44 | 48 | 22 | 1056 | 263.9234 | 5944.5703 | 0
53 45 | 21 | 32 | 672 | 422.42358 | 5235.915 | 0
54 46 | 5 | 18 | 90 | 302.97906 | 13293.447 | 0
55 47 | 12 | 24 | 288 | 325.6916 | 4084.442 | 0
56 48 | 72 | 13 | 936 | 360.8169 | 7499.508 | 0
57 49 | 43 | 24 | 1032 | 265.93164 | 6009.701 | 0
58 50 | 14 | 31 | 434 | 423.7791 | 4236.278 | 0
59 51 | 43 | 14 | 602 | 364.65057 | 6938.7554 | 0
60 52 | 9 | 21 | 189 | 397.09445 | 8630.353 | 0
61 53 | 50 | 15 | 750 | 288.08157 | 5100.588 | 0
62 54 | 28 | 19 | 532 | 397.55252 | 8264.971 | 0
63 55 | 2 | 37 | 74 | 257.47296 | 12608.535 | 0
64 56 | 37 | 20 | 740 | 393.25366 | 6683.926 | 0
65 57 | 20 | 19 | 380 | 411.1708 | 6788.7314 | 0
66 58 | 69 | 19 | 1311 | 309.80255 | 5961.776 | 0
67 59 | 52 | 27 | 1404 | 320.31512 | 4769.724 | 0
68 60 | 39 | 18 | 702 | 409.38147 | 7566.9824 | 0
69 61 | 21 | 18 | 378 | 430.08206 | 7622.618 | 0
70 62 | 43 | 25 | 1075 | 380.73218 | 5320.311 | 0
71 63 | 18 | 31 | 558 | 385.48874 | 3891.026 | 0
72 64 | 32 | 20 | 640 | 394.38077 | 7474.1694 | 0
73 65 | 44 | 14 | 616 | 361.95868 | 6896.776 | 0
74 66 | 47 | 15 | 705 | 294.42923 | 3419.358 | 0
75 67 | 27 | 25 | 675 | 429.26218 | 5316.536 | 0
76 68 | 47 | 24 | 1128 | 390.02103 | 5590.9233 | 0
77 69 | 18 | 16 | 288 | 380.61258 | 8707.474 | 0
78 70 | 22 | 29 | 638 | 416.0425 | 4532.3687 | 0
79 71 | 2 | 52 | 104 | 184.8413 | 9455.265 | 0
80 72 | 43 | 18 | 774 | 391.97424 | 6221.4204 | 0
81 73 | 43 | 35 | 1505 | 399.86578 | 4363.5215 | 0
82 74 | 39 | 16 | 624 | 425.50522 | 9422.357 | 0
83 75 | 17 | 26 | 442 | 428.31274 | 6103.0566 | 0
84 76 | 28 | 22 | 616 | 340.24344 | 5377.256 | 0
85 77 | 45 | 19 | 855 | 289.4797 | 6098.8125 | 0
86 78 | 2 | 49 | 98 | 66.27381 | 12810.174 | 0

```
 87   79 | 33 | 23 | 759 | 358.46017 | 5427.3643 | 0
 88   80 | 7 | 28 | 196 | 414.87424 | 5232.352 | 0
 89   81 | 32 | 22 | 704 | 362.89316 | 5548.298 | 0
 90   82 | 11 | 20 | 220 | 409.43295 | 5017.3857 | 0
 91   83 | 21 | 11 | 231 | 393.1678 | 12809.008 | 0
 92   84 | 6 | 41 | 246 | 329.3846 | 4914.534 | 0
 93   85 | 4 | 30 | 120 | 352.9455 | 10079.1 | 0
 94   86 | 24 | 20 | 480 | 420.54633 | 7422.1426 | 0
 95   87 | 35 | 33 | 1155 | 334.49033 | 4689.2886 | 0
 96   88 | 36 | 24 | 864 | 296.29758 | 1939.2076 | 0
 97   89 | 2 | 31 | 62 | 292.62488 | 9264.893 | 0
 98   90 | 15 | 32 | 480 | 430.56433 | 4325.621 | 0
 99   91 | 26 | 24 | 624 | 383.32858 | 6530.718 | 0
100   92 | 36 | 28 | 1008 | 431.7446 | 4830.9297 | 0
101   93 | 20 | 19 | 380 | 398.24777 | 9217.031 | 0
102   94 | 21 | 19 | 399 | 373.25732 | 6345.0757 | 0
103   95 | 33 | 43 | 1419 | 352.1744 | 3374.9387 | 0
104   96 | 2 | 43 | 86 | 259.84174 | 6977.2466 | 0
105   97 | 11 | 25 | 275 | 404.49243 | 6474.787 | 0
106   98 | 33 | 22 | 726 | 357.80273 | 6191.2905 | 0
107   99 | 3 | 30 | 90 | 345.9133 | 7232.6445 | 0
108   100 | 29 | 22 | 638 | 388.9521 | 3320.1787 | 0
```

Annexe E

Résultats SMOB et implémentations

E.1 Yeast dataset

E.1.1 Recherche locale sur la population finale : Approche Lamarckienne classique

```
result_yeast_08042010015113.txt         Temps total d'execution : 1091 sec
Coverage: 0.58099455
Overlapping: 0.635703
Genes: 0.64979196
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 68.5 | 12.52 | 864.2 | 226.86484 | 1230.7078 | 1.7541035
Ecart type | 65.6077 | 1.813725 | 857.3559 | 26.399235 | 752.3203 | 1.6941088
1 | 187 | 12 | 2244 | 214.68625 | 770.0251 | 0.12791292
2 | 99 | 15 | 1485 | 234.26805 | 699.69794 | 0.3630157
3 | 196 | 10 | 1960 | 220.08461 | 754.2733 | 0.6102129
4 | 35 | 14 | 490 | 273.3369 | 1357.6418 | 0.1475848
5 | 21 | 13 | 273 | 236.95287 | 1375.3474 | 0.21995349
6 | 71 | 12 | 852 | 253.77563 | 895.40735 | 0.27496043
7 | 89 | 16 | 1424 | 211.71573 | 676.08374 | 0.07467381
8 | 21 | 12 | 252 | 264.66946 | 2063.9812 | 0.0672445
9 | 50 | 11 | 550 | 260.44937 | 1345.9178 | 0.17719656
10 | 29 | 12 | 348 | 222.66093 | 1057.1456 | 0.40915722
11 | 54 | 13 | 702 | 260.12064 | 1290.1938 | 0.13262953
12 | 25 | 12 | 300 | 224.07883 | 3025.8286 | 0.18635738
13 | 228 | 13 | 2964 | 213.75114 | 604.2093 | 1.1704046
14 | 26 | 15 | 390 | 262.21805 | 949.5318 | 0.5271488
15 | 172 | 13 | 2236 | 220.98422 | 622.77167 | 1.3273288
```

16	162	11	1782	200.1125	799.7939	1.6710546
17	192	15	2880	210.35628	441.69366	1.7547355
18	153	15	2295	216.02928	439.36993	2.1929057
19	21	13	273	233.16637	1597.6522	0.71076536
20	77	13	1001	257.75604	973.19055	0.98968744
21	127	10	1270	210.01439	650.4662	1.4865106
22	16	13	208	262.9313	1989.7318	0.19295841
23	146	11	1606	227.34486	694.6785	1.5548841
24	13	14	182	223.6506	1330.0059	0.91151154
25	201	13	2613	203.67516	526.19806	2.0334425
26	30	10	300	229.98228	1419.2717	0.36744347
27	17	13	221	242.56816	1446.8798	0.29948995
28	167	14	2338	218.56837	479.79535	2.8790784
29	22	14	308	262.16855	1093.5126	0.66575956
30	32	12	384	233.0782	1169.7485	0.5077494
31	51	12	612	245.40317	1013.03827	1.0749145
32	210	14	2940	220.72006	390.6698	2.2940857
33	30	11	330	221.16599	1731.5446	1.1822449
34	11	9	99	225.62756	3308.7285	0.3112596
35	32	12	384	265.21875	1296.0653	0.885707
36	28	16	448	265.36234	1011.7868	1.6063939
37	38	14	532	261.81128	1262.7776	0.6101743
38	32	10	320	237.23672	1654.531	1.0320616
39	25	10	250	266.29855	1490.6921	0.46673965
40	7	12	84	229.59128	1889.8346	0.63958776
41	6	12	72	207.24536	3114.118	0.14972359
42	134	16	2144	200.50154	470.0492	4.5128903
43	156	11	1716	175.24432	750.7356	3.6741993
44	132	12	1584	230.5565	674.4992	2.4154952
45	36	9	324	156.60881	1659.1808	1.2598398
46	241	12	2892	192.54755	496.3405	3.4648213
47	176	14	2464	203.7236	388.11673	4.34603
48	19	12	228	258.01007	1734.8073	0.82165265
49	66	9	594	182.0455	1007.96265	3.0217977
50	33	12	396	252.29155	1170.3744	1.6017921
51	176	13	2288	204.15567	469.37827	4.268757
52	64	12	768	239.1758	841.86334	1.2241372
53	122	9	1098	207.79167	841.38745	4.7333417
54	51	14	714	254.74828	767.02313	1.2318969
55	81	14	1134	193.20279	771.02405	5.7254825
56	23	10	230	277.53897	1510.7162	0.6750566
57	62	11	682	232.62216	1389.273	1.3823088
58	33	13	429	252.29143	1112.6995	1.3994086
59	171	15	2565	179.95755	333.02792	4.0476694

60	125	12	1500	230.09445	658.68823	3.5570228
61	5	14	70	194.18782	1321.226	0.8294766
62	15	10	150	226.14288	1934.5652	0.60293144
63	3	11	33	216.60237	3170.2036	0.07892215
64	11	14	154	241.36475	1685.7174	1.0615414
65	14	15	210	251.63377	1105.3813	0.94915986
66	24	12	288	219.86237	743.4065	1.172997
67	15	14	210	242.63313	1093.4437	1.5839925
68	4	16	64	189.69727	4786.376	0.4647394
69	6	16	96	279.059	2260.8782	0.15855713
70	4	14	56	231.53406	2357.2363	0.636371
71	121	14	1694	201.99728	629.6028	0.75289243
72	162	12	1944	188.01149	492.3432	6.36658
73	33	10	330	240.90775	1166.3862	2.7360342
74	135	12	1620	179.80807	682.4798	5.052203
75	15	11	165	218.3907	1884.3208	1.3281821
76	2	15	30	186.85553	2416.2844	0.4576031
77	7	14	98	227.54745	1086.6488	0.1948605
78	28	13	364	260.45895	2679.8132	0.9983886
79	61	16	976	207.17834	485.83005	4.998356
80	47	12	564	213.01303	915.254	2.179012
81	164	14	2296	194.96805	353.8579	5.5962744
82	34	12	408	223.84871	814.0382	1.7697695
83	26	11	286	218.93254	977.61285	0.85460263
84	43	14	602	234.9	758.25397	2.2220128
85	27	11	297	234.82457	1731.1354	1.8542254
86	21	9	189	244.06815	1881.2318	0.64511496
87	141	12	1692	204.00456	578.3078	5.8887377
88	28	10	280	252.26054	1960.4415	0.9667921
89	16	15	240	254.72026	815.9267	1.3374436
90	58	11	638	218.13434	957.51056	2.8700912
91	28	11	308	194.99464	1030.924	3.9012198
92	168	13	2184	190.27888	417.73306	6.460288
93	5	13	65	209.47977	2284.6743	1.4832522
94	21	11	231	198.32216	967.85004	3.262218
95	26	11	286	257.41574	1331.206	1.5226072
96	22	12	264	232.99452	1163.481	1.3200694
97	7	15	105	273.4423	1281.9717	0.93831545
98	182	12	2184	195.34715	568.0833	7.425554
99	63	11	693	208.70448	727.4564	4.1481485
100	8	13	104	254.01181	1820.729	0.6906144

E.1.2 Recherche locale sur la population finale : Approche Lamarckienne modifiée

```
result_yeast_08042010120929.txt         Temps total d'execution : 960 sec
Coverage: 0.57764953
Overlapping: 0.61533844
Genes: 0.6688627
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 86.18 | 12.03 | 1019.11 | 187.17946 | 1001.3986 | 0
Ecart type | 86.06815 | 2.1375453 | 983.94995 | 41.948284 | 701.89276 | 0
1 | 182 | 12 | 2184 | 216.25981 | 726.1835 | 0
2 | 59 | 13 | 767 | 245.72011 | 965.8828 | 0
3 | 191 | 14 | 2674 | 214.38576 | 630.3879 | 0
4 | 184 | 11 | 2024 | 188.39796 | 593.37244 | 0
5 | 168 | 12 | 2016 | 195.16664 | 712.40845 | 0
6 | 139 | 9 | 1251 | 179.61264 | 795.4112 | 0
7 | 47 | 11 | 517 | 242.84369 | 924.8235 | 0
8 | 189 | 13 | 2457 | 207.55302 | 452.89636 | 0
9 | 157 | 12 | 1884 | 214.30598 | 576.6845 | 0
10 | 80 | 11 | 880 | 215.30162 | 905.5551 | 0
11 | 197 | 12 | 2364 | 224.09975 | 506.61728 | 0
12 | 8 | 14 | 112 | 175.88606 | 2035.6028 | 0
13 | 7 | 12 | 84 | 161.17374 | 2208.4622 | 0
14 | 24 | 14 | 336 | 220.48337 | 1249.5409 | 0
15 | 40 | 11 | 440 | 229.13368 | 1448.136 | 0
16 | 38 | 11 | 418 | 225.26033 | 1306.2407 | 0
17 | 58 | 11 | 638 | 223.63179 | 1133.4579 | 0
18 | 175 | 10 | 1750 | 170.33165 | 616.0534 | 0
19 | 3 | 10 | 30 | 130.9911 | 3656.24 | 0
20 | 14 | 5 | 70 | 46.582066 | 632.7373 | 0
21 | 71 | 11 | 781 | 217.34512 | 917.54785 | 0
22 | 29 | 9 | 261 | 176.73163 | 2567.5222 | 0
23 | 368 | 7 | 2576 | 104.38617 | 147.74796 | 0
24 | 182 | 11 | 2002 | 171.28531 | 570.9807 | 0
25 | 57 | 14 | 798 | 242.17616 | 855.3449 | 0
26 | 75 | 10 | 750 | 203.93094 | 773.06006 | 0
27 | 48 | 12 | 576 | 216.20575 | 889.77625 | 0
28 | 132 | 15 | 1980 | 195.41014 | 489.7119 | 0
29 | 162 | 9 | 1458 | 192.80168 | 565.50195 | 0
30 | 26 | 13 | 338 | 236.53023 | 1476.9541 | 0
31 | 43 | 12 | 516 | 176.8099 | 861.83527 | 0
32 | 159 | 13 | 2067 | 181.23352 | 384.18265 | 0
33 | 157 | 10 | 1570 | 175.94427 | 519.39 | 0
```

```
42   34 | 161 | 11 | 1771 | 183.39825 | 666.31213 | 0
43   35 | 3 | 14 | 42 | 102.778915 | 668.3691 | 0
44   36 | 132 | 13 | 1716 | 178.38985 | 399.28918 | 0
45   37 | 85 | 17 | 1445 | 201.41116 | 656.7152 | 0
46   38 | 10 | 11 | 110 | 156.85687 | 2199.878 | 0
47   39 | 122 | 14 | 1708 | 171.78096 | 389.47772 | 0
48   40 | 16 | 13 | 208 | 218.04199 | 2493.826 | 0
49   41 | 375 | 11 | 4125 | 166.56432 | 272.6039 | 0
50   42 | 34 | 14 | 476 | 225.90372 | 903.2618 | 0
51   43 | 23 | 13 | 299 | 225.31511 | 882.32965 | 0
52   44 | 137 | 15 | 2055 | 209.12239 | 462.82245 | 0
53   45 | 57 | 12 | 684 | 126.62001 | 311.28384 | 0
54   46 | 4 | 10 | 40 | 60.346844 | 179.8875 | 0
55   47 | 6 | 15 | 90 | 193.06496 | 2970.015 | 0
56   48 | 164 | 12 | 1968 | 176.85828 | 439.05942 | 0
57   49 | 10 | 14 | 140 | 194.56203 | 1323.949 | 0
58   50 | 90 | 10 | 900 | 200.35873 | 618.9664 | 0
59   51 | 16 | 13 | 208 | 177.47125 | 2434.4177 | 0
60   52 | 34 | 12 | 408 | 231.78291 | 1144.5557 | 0
61   53 | 82 | 11 | 902 | 218.93146 | 847.9816 | 0
62   54 | 182 | 15 | 2730 | 192.14491 | 334.13983 | 0
63   55 | 175 | 14 | 2450 | 211.90382 | 423.96246 | 0
64   56 | 222 | 15 | 3330 | 200.69714 | 361.18494 | 0
65   57 | 511 | 11 | 5621 | 170.0861 | 269.27454 | 0
66   58 | 27 | 13 | 351 | 239.27713 | 1094.5193 | 0
67   59 | 94 | 11 | 1034 | 158.31827 | 627.43823 | 0
68   60 | 2 | 17 | 34 | 108.88062 | 1659.72 | 0
69   61 | 42 | 9 | 378 | 171.14787 | 1035.6853 | 0
70   62 | 19 | 11 | 209 | 206.27841 | 1850.6385 | 0
71   63 | 100 | 10 | 1000 | 168.1667 | 940.6796 | 0
72   64 | 54 | 11 | 594 | 198.4378 | 681.95123 | 0
73   65 | 41 | 14 | 574 | 233.27611 | 946.07916 | 0
74   66 | 153 | 11 | 1683 | 147.64891 | 223.62366 | 0
75   67 | 10 | 12 | 120 | 209.38077 | 1697.9382 | 0
76   68 | 31 | 9 | 279 | 160.17781 | 2241.8337 | 0
77   69 | 131 | 15 | 1965 | 200.1321 | 304.27826 | 0
78   70 | 95 | 10 | 950 | 171.45712 | 804.38934 | 0
79   71 | 71 | 12 | 852 | 171.88254 | 635.8884 | 0
80   72 | 174 | 11 | 1914 | 166.93288 | 441.1326 | 0
81   73 | 182 | 11 | 2002 | 174.81424 | 552.1292 | 0
82   74 | 15 | 13 | 195 | 206.27698 | 2338.0488 | 0
83   75 | 10 | 11 | 110 | 156.86938 | 2197.2087 | 0
84   76 | 25 | 11 | 275 | 221.81262 | 1515.6299 | 0
85   77 | 57 | 14 | 798 | 212.42749 | 570.6147 | 0
```

```
 86   78 | 49 | 11 | 539 | 217.40407 | 765.9404 | 0
 87   79 | 87 | 14 | 1218 | 191.19922 | 551.6756 | 0
 88   80 | 39 | 14 | 546 | 238.59445 | 1019.5639 | 0
 89   81 | 38 | 15 | 570 | 229.51451 | 587.52747 | 0
 90   82 | 33 | 13 | 429 | 214.91977 | 1324.6384 | 0
 91   83 | 33 | 12 | 396 | 209.65794 | 962.88434 | 0
 92   84 | 23 | 13 | 299 | 221.24982 | 1412.7538 | 0
 93   85 | 101 | 16 | 1616 | 190.33067 | 400.13702 | 0
 94   86 | 2 | 16 | 32 | 77.52246 | 1090.0137 | 0
 95   87 | 67 | 11 | 737 | 187.21915 | 518.98175 | 0
 96   88 | 39 | 13 | 507 | 238.9722 | 1061.5784 | 0
 97   89 | 17 | 9 | 153 | 221.46844 | 2315.6462 | 0
 98   90 | 27 | 10 | 270 | 214.96431 | 1457.0713 | 0
 99   91 | 100 | 11 | 1100 | 199.27519 | 669.17096 | 0
100   92 | 2 | 13 | 26 | 82.597626 | 1652.3436 | 0
101   93 | 133 | 12 | 1596 | 207.77336 | 464.5686 | 0
102   94 | 12 | 14 | 168 | 218.92908 | 903.6996 | 0
103   95 | 150 | 12 | 1800 | 160.24323 | 228.83813 | 0
104   96 | 15 | 14 | 210 | 178.89497 | 977.72925 | 0
105   97 | 32 | 6 | 192 | 43.028103 | 102.93746 | 0
106   98 | 3 | 11 | 33 | 132.83563 | 2645.4717 | 0
107   99 | 24 | 9 | 216 | 169.65263 | 1645.1517 | 0
108   100 | 139 | 14 | 1946 | 175.79247 | 272.29892 | 0
```

E.1.3 Recherche locale à chaque génération

```
1  result_yeast_08042010104645.txt        Temps total d'execution : 858 sec
2  Coverage: 0.5591091
3  Overlapping: 0.39687
4  Genes: 0.7552011
5  Conditions: 1.0
6  Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
7  Moyenne | 42.28 | 10.26 | 436.65 | 278.61157 | 1225.3528 | 0.64050406
8  Ecart type | 18.989511 | 1.3828952 | 208.60757 | 24.782097 | 553.8407 | 0.31735423
9  1 | 54 | 11 | 594 | 290.72324 | 1333.4928 | 0.10753976
10 2 | 32 | 12 | 384 | 281.48965 | 1448.8105 | 0.14233118
11 3 | 82 | 9 | 738 | 286.5083 | 1199.1816 | 0.18783663
12 4 | 72 | 11 | 792 | 295.23395 | 1104.7137 | 0.30433586
13 5 | 69 | 13 | 897 | 289.58298 | 1101.1918 | 0.37207922
14 6 | 83 | 9 | 747 | 288.1674 | 1319.5614 | 0.49230534
15 7 | 78 | 10 | 780 | 291.6083 | 1092.0548 | 0.5244611
16 8 | 31 | 9 | 279 | 278.44232 | 1742.6406 | 0.14849567
17 9 | 72 | 9 | 648 | 293.56073 | 1226.6625 | 0.6196097
18 10 | 56 | 11 | 616 | 291.03552 | 1085.7339 | 0.37229228
19 11 | 55 | 11 | 605 | 294.63605 | 1191.3538 | 0.69738054
20 12 | 60 | 9 | 540 | 287.4899 | 1124.1456 | 0.38984433
21 13 | 25 | 10 | 250 | 237.13768 | 3495.6938 | 0.22592038
22 14 | 36 | 9 | 324 | 287.17255 | 1675.1782 | 0.4972042
23 15 | 65 | 10 | 650 | 290.30475 | 1031.7571 | 0.5965852
24 16 | 62 | 9 | 558 | 285.1835 | 1092.1245 | 0.59090364
25 17 | 66 | 10 | 660 | 289.06772 | 1002.5326 | 0.59339553
26 18 | 32 | 10 | 320 | 284.05597 | 1862.5381 | 0.32096517
27 19 | 37 | 9 | 333 | 283.39047 | 1760.9567 | 0.43662202
28 20 | 44 | 11 | 484 | 294.75296 | 969.4346 | 0.4263199
29 21 | 39 | 11 | 429 | 274.19254 | 1223.8718 | 0.4514193
30 22 | 24 | 9 | 216 | 262.10535 | 1244.7795 | 0.2538826
31 23 | 48 | 9 | 432 | 285.3722 | 1104.0106 | 0.4978196
32 24 | 40 | 13 | 520 | 288.82983 | 954.231 | 0.46962065
33 25 | 38 | 10 | 380 | 282.18167 | 1738.9102 | 0.27290228
34 26 | 17 | 9 | 153 | 273.683 | 1425.9514 | 0.2629813
35 27 | 20 | 9 | 180 | 260.9604 | 1695.7147 | 0.5879186
36 28 | 49 | 10 | 490 | 289.0655 | 1049.2018 | 0.56733406
37 29 | 56 | 9 | 504 | 292.40668 | 1295.1652 | 0.56601787
38 30 | 78 | 13 | 1014 | 294.9848 | 848.67365 | 0.89199704
39 31 | 59 | 12 | 708 | 293.99744 | 965.9254 | 0.5204138
40 32 | 5 | 10 | 50 | 226.24156 | 2546.0562 | 0.15606844
41 33 | 39 | 12 | 468 | 258.1899 | 1077.4731 | 0.12980372
42 34 | 49 | 13 | 637 | 282.68094 | 691.32825 | 0.39390868
```

35	40	13	520	297.05212	962.4139	0.6465594
36	6	11	66	218.14049	3465.4624	0.113194905
37	66	11	726	295.6924	1025.8954	0.9698482
38	4	10	40	124.531876	2946.6323	0.06557421
39	68	12	816	290.67514	917.8112	0.78924644
40	52	9	468	290.9744	1166.9471	0.7192703
41	56	9	504	284.62653	1254.186	0.9964251
42	38	9	342	285.86752	940.73535	0.44605613
43	64	9	576	284.8689	1086.1399	1.1379304
44	28	9	252	282.1111	1155.666	0.5216127
45	62	9	558	293.9164	948.5385	0.96827316
46	77	9	693	285.21698	983.9607	1.1169758
47	31	12	372	263.6077	1416.5338	0.50787604
48	8	11	88	228.5328	2580.2888	0.2243005
49	19	10	190	261.1137	2132.357	0.48806182
50	19	9	171	244.23033	2856.938	0.6062627
51	37	9	333	285.79483	1339.131	0.598251
52	40	9	360	289.19333	1224.6057	0.55872494
53	50	12	600	292.99426	961.6254	0.6781291
54	32	11	352	290.32773	1057.6777	0.3978315
55	19	9	171	245.7881	1121.3418	0.52471685
56	53	9	477	295.65613	939.6954	0.8541717
57	27	9	243	281.67966	893.75916	0.47593668
58	39	9	351	288.30716	857.36566	0.6679878
59	50	11	550	289.9769	866.03656	0.92896795
60	24	10	240	284.46722	1342.9891	0.8224281
61	46	12	552	292.37592	883.6679	0.9879899
62	28	10	280	284.7173	871.6069	0.5131702
63	52	9	468	283.33112	1153.5144	1.3406869
64	33	10	330	295.03696	1093.1644	0.6730627
65	41	15	615	274.32565	900.10114	0.5520124
66	39	9	351	281.27112	964.51807	0.7340717
67	30	9	270	286.16235	1461.5537	0.92660475
68	39	10	390	271.4515	1033.299	0.8066734
69	53	10	530	287.8325	901.2023	0.8030636
70	21	10	210	286.83643	940.24255	0.4282088
71	30	12	360	285.65472	580.0661	0.3661266
72	70	9	630	291.3783	959.0504	0.94983995
73	25	10	250	283.13693	950.3655	0.60292536
74	55	9	495	291.38727	903.1415	1.0129277
75	30	9	270	290.55054	1404.5996	0.9828629
76	35	12	420	276.99066	952.8759	0.8208589
77	43	12	516	290.22308	985.74396	0.9754584
78	69	9	621	285.1287	1023.0698	1.2054468

```
 87   79 | 35 | 11 | 385 | 289.03152 | 979.7588 | 0.85891986
 88   80 | 50 | 12 | 600 | 282.46298 | 755.2444 | 1.0358367
 89   81 | 45 | 11 | 495 | 285.1274 | 972.6767 | 1.2457075
 90   82 | 63 | 11 | 693 | 292.81464 | 796.9633 | 1.1895719
 91   83 | 47 | 13 | 611 | 289.75107 | 677.7953 | 1.0270966
 92   84 | 45 | 12 | 540 | 297.65884 | 973.3538 | 1.0188211
 93   85 | 68 | 13 | 884 | 289.47067 | 745.9669 | 1.6021166
 94   86 | 48 | 9 | 432 | 290.84152 | 756.57916 | 0.56063145
 95   87 | 42 | 9 | 378 | 270.4051 | 845.9519 | 0.7822491
 96   88 | 42 | 11 | 462 | 296.30975 | 630.4454 | 0.8122681
 97   89 | 44 | 12 | 528 | 288.2098 | 774.6815 | 0.7714233
 98   90 | 11 | 9 | 99 | 240.43468 | 1636.3617 | 0.28573743
 99   91 | 12 | 10 | 120 | 279.7878 | 1054.5199 | 0.5298187
100   92 | 4 | 10 | 40 | 180.45753 | 2353.5352 | 0.036086902
101   93 | 57 | 9 | 513 | 277.25485 | 796.5484 | 1.0479994
102   94 | 52 | 11 | 572 | 292.85568 | 830.0545 | 1.2101521
103   95 | 23 | 9 | 207 | 276.80112 | 781.38214 | 0.78304315
104   96 | 37 | 9 | 333 | 277.1529 | 993.7435 | 0.9571739
105   97 | 28 | 9 | 252 | 265.85962 | 1276.4514 | 0.82483137
106   98 | 20 | 10 | 200 | 288.413 | 624.9464 | 0.5508015
107   99 | 26 | 9 | 234 | 286.82657 | 1102.228 | 0.9487661
108   100 | 9 | 10 | 90 | 225.66591 | 2048.522 | 0.39624095
```

E.1.4 Opérateurs génétiques intelligents : 25 essais

```
result_yeast_01022010113927.txt         Temps total d'execution : 1281 sec
Coverage: 0.5510933
Overlapping: 0.53928715
Genes: 0.69348127
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 55.82 | 9.93 | 553.2 | 219.99387 | 1348.1538 | 1.124998
Ecart type | 16.930079 | 1.193775 | 173.53674 | 53.362686 | 763.3071 | 0.58132875
1 | 73 | 9 | 657 | 150.9557 | 1224.0333 | 0.40329865
2 | 48 | 13 | 624 | 297.1245 | 1337.9377 | 0.35623568
3 | 43 | 12 | 516 | 297.40677 | 3204.7112 | 0.12373681
4 | 63 | 10 | 630 | 144.31377 | 918.41864 | 0.43453228
5 | 47 | 10 | 470 | 299.21536 | 1864.8313 | 0.20112506
6 | 79 | 9 | 711 | 216.65935 | 1305.9839 | 0.26406175
7 | 68 | 11 | 748 | 285.4915 | 1636.3224 | 0.25036812
8 | 64 | 10 | 640 | 216.65439 | 1247.9944 | 0.4341591
9 | 28 | 10 | 280 | 298.41092 | 2347.431 | 0.139488
10 | 87 | 9 | 783 | 182.81963 | 994.3827 | 0.5523555
11 | 65 | 11 | 715 | 283.78647 | 1489.2557 | 0.2306019
12 | 68 | 9 | 612 | 155.45753 | 1066.8131 | 0.8295591
13 | 65 | 10 | 650 | 198.09123 | 786.2885 | 0.7371305
14 | 57 | 10 | 570 | 233.923 | 1316.0701 | 0.9806471
15 | 66 | 9 | 594 | 188.35428 | 1054.5647 | 0.470609
16 | 57 | 9 | 513 | 174.23036 | 863.2084 | 0.7153939
17 | 97 | 9 | 873 | 133.22514 | 833.97253 | 0.9765302
18 | 46 | 10 | 460 | 286.11746 | 2040.2437 | 0.40172458
19 | 60 | 10 | 600 | 261.64813 | 1364.7571 | 1.023301
20 | 65 | 12 | 780 | 275.88052 | 1668.6984 | 0.42243177
21 | 62 | 9 | 558 | 192.37596 | 955.6576 | 0.7078444
22 | 64 | 9 | 576 | 184.48392 | 890.2132 | 1.1167396
23 | 70 | 9 | 630 | 136.52943 | 1008.54346 | 0.24363409
24 | 61 | 9 | 549 | 232.40941 | 1444.1593 | 0.8546718
25 | 57 | 9 | 513 | 251.47223 | 1893.4906 | 0.8456209
26 | 74 | 9 | 666 | 193.48178 | 994.3701 | 1.0171555
27 | 80 | 10 | 800 | 163.28023 | 742.5552 | 1.3408513
28 | 58 | 9 | 522 | 152.38094 | 893.4434 | 1.2865981
29 | 54 | 13 | 702 | 274.74915 | 1253.0002 | 0.7552999
30 | 67 | 10 | 670 | 110.358154 | 745.7709 | 1.5926731
31 | 43 | 9 | 387 | 287.45398 | 2048.5764 | 0.3516767
32 | 16 | 9 | 144 | 269.44733 | 3983.8052 | 0.2223347
33 | 66 | 9 | 594 | 254.68097 | 1398.5549 | 0.78772473
34 | 74 | 9 | 666 | 210.64748 | 1087.9834 | 1.2818857
```

35	22	9	198	299.3631	2965.0327	0.25724986
36	74	9	666	210.536	1423.2739	0.87714225
37	82	9	738	128.28673	806.7616	1.5761241
38	70	10	700	233.68138	1118.5902	0.9969238
39	35	11	385	288.74176	3057.8179	1.150207
40	51	11	561	287.31372	878.55927	0.86761564
41	62	11	682	242.91624	1059.2223	1.6022317
42	58	11	638	158.0798	714.3047	1.3553989
43	59	10	590	182.15688	848.108	1.0124373
44	31	9	279	289.77823	3145.6753	0.5109917
45	51	11	561	231.31602	972.7816	1.2986947
46	97	9	873	122.50918	610.6448	1.5870513
47	21	9	189	291.41812	2933.1626	0.5339926
48	64	9	576	257.1765	1040.3794	0.39931232
49	58	9	522	219.29393	1312.2057	1.3836284
50	38	9	342	287.88528	2234.9736	1.1305654
51	24	9	216	279.41226	3039.4375	0.48761487
52	43	9	387	199.66031	1057.9165	1.3303189
53	71	12	852	150.0606	580.65826	1.5175713
54	66	9	594	162.88818	818.4472	1.8635086
55	46	10	460	198.34468	816.8281	1.0188584
56	48	11	528	183.64473	700.06287	1.3684955
57	65	11	715	280.1936	1481.1241	1.3371186
58	55	9	495	160.86008	869.57306	1.8613125
59	64	9	576	239.1347	1244.9534	1.4413416
60	33	9	297	299.065	2729.091	0.7587495
61	61	9	549	199.90723	1059.296	1.2359762
62	46	9	414	231.43768	1500.8761	1.166312
63	43	9	387	296.8247	1793.1451	1.0417988
64	38	9	342	288.89648	2360.0725	1.1699994
65	12	11	132	298.24643	3267.4937	0.8988769
66	62	12	744	194.10738	590.38196	1.6785948
67	62	9	558	244.49188	1448.8263	1.1994246
68	68	10	680	224.33336	1204.1204	1.3794665
69	10	10	100	298.86957	4324.817	0.19792221
70	47	9	423	209.0296	1073.0133	0.88506734
71	78	11	858	139.52313	821.5778	0.8667462
72	71	9	639	193.96515	1010.86774	2.5536122
73	46	10	460	250.73628	1183.7914	1.79393
74	38	12	456	211.20137	930.11566	1.780134
75	57	11	627	218.1266	997.8199	1.3395591
76	46	11	506	236.95479	942.8208	1.3482755
77	62	9	558	189.35068	875.156	1.7862974
78	45	11	495	248.48145	1051.748	1.6946214

```
87   79 | 60 | 11 | 660 | 156.28539 | 678.92 | 2.484987
88   80 | 68 | 10 | 680 | 254.08017 | 1197.9518 | 1.539222
89   81 | 54 | 11 | 594 | 217.44617 | 832.8177 | 1.1623253
90   82 | 75 | 9 | 675 | 147.67549 | 825.2391 | 2.3869197
91   83 | 49 | 9 | 441 | 222.00261 | 1542.2656 | 1.1487216
92   84 | 64 | 9 | 576 | 127.21324 | 678.2539 | 2.3647587
93   85 | 44 | 9 | 396 | 185.53537 | 1030.0505 | 1.3515869
94   86 | 62 | 10 | 620 | 200.75969 | 778.4823 | 1.6218761
95   87 | 49 | 13 | 637 | 235.64616 | 859.1994 | 1.2739943
96   88 | 65 | 9 | 585 | 193.35435 | 732.0001 | 1.5952625
97   89 | 32 | 9 | 288 | 248.51454 | 1552.4628 | 2.0960557
98   90 | 67 | 9 | 603 | 160.22188 | 704.55994 | 1.9663595
99   91 | 37 | 9 | 333 | 279.6867 | 1376.7968 | 1.3169178
100  92 | 37 | 9 | 333 | 255.28186 | 1732.4728 | 1.2589922
101  93 | 65 | 10 | 650 | 157.61845 | 739.819 | 2.185849
102  94 | 42 | 12 | 504 | 252.02687 | 968.16284 | 1.1591132
103  95 | 36 | 10 | 360 | 223.4854 | 1349.8643 | 1.8102965
104  96 | 80 | 12 | 960 | 125.47253 | 438.30075 | 2.2188265
105  97 | 52 | 9 | 468 | 202.04863 | 1037.9882 | 1.2128233
106  98 | 46 | 14 | 644 | 208.90947 | 678.8877 | 2.0466506
107  99 | 50 | 9 | 450 | 264.18527 | 1631.6998 | 1.2239989
108  100 | 76 | 12 | 912 | 130.21735 | 647.5754 | 1.1811943
```

E.1.5 Opérateurs génétiques intelligents : 50 essais

```
result_yeast_02022010122129.txt         Temps total d'execution : 1672 sec
Coverage: 0.4959207
Overlapping: 0.48173892
Genes: 0.66539526
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 48.74 | 9.28 | 448.02 | 199.65746 | 1637.7805 | 3.40E+36
Ecart type | 16.72281 | 0.64930725 | 150.56049 | 60.088337 | 1372.4622 |  Infinity
1 | 68 | 9 | 612 | 147.44017 | 1266.8219 | 0.30463442
2 | 46 | 10 | 460 | 194.19228 | 1155.131 | 0.20275193
3 | 59 | 9 | 531 | 197.71648 | 1822.8907 | 0.17862366
4 | 53 | 10 | 530 | 206.47166 | 1396.624 | 0.32121962
5 | 69 | 9 | 621 | 180.89 | 1033.548 | 0.43969303
6 | 70 | 9 | 630 | 113.72315 | 970.94324 | 1.0735431
7 | 48 | 9 | 432 | 229.64906 | 1754.7101 | 0.53716683
8 | 28 | 9 | 252 | 299.29565 | 3100.4802 | 0.18075149
9 | 49 | 10 | 490 | 297.64337 | 2247.8716 | 0.43308425
10 | 74 | 9 | 666 | 139.43604 | 830.31177 | 0.6080784
11 | 49 | 9 | 441 | 215.49626 | 2026.5367 | 0.22731678
12 | 63 | 9 | 567 | 137.70763 | 837.9942 | 0.8539863
13 | 46 | 9 | 414 | 251.48082 | 1848.1466 | 0.56413096
14 | 52 | 10 | 520 | 254.09033 | 1722.5972 | 0.31121296
15 | 52 | 9 | 468 | 211.58485 | 1220.4633 | 0.49764645
16 | 50 | 9 | 450 | 216.41028 | 1779.6158 | 0.5995255
17 | 55 | 9 | 495 | 266.43314 | 1387.5695 | 0.4433154
18 | 47 | 11 | 517 | 197.42735 | 1693.9613 | 0.7038983
19 | 29 | 9 | 261 | 275.5259 | 2215.4983 | 0.36359152
20 | 55 | 9 | 495 | 243.24146 | 1599.7852 | 0.40620548
21 | 62 | 9 | 558 | 199.7859 | 1295.26 | 0.5661288
22 | 63 | 9 | 567 | 172.9549 | 1087.4849 | 1.0873988
23 | 66 | 9 | 594 | 160.9244 | 886.7499 | 0.7486348
24 | 73 | 9 | 657 | 153.94649 | 836.9651 | 1.0358908
25 | 51 | 9 | 459 | 220.11624 | 1640.4984 | 0.39891386
26 | 71 | 9 | 639 | 188.93013 | 1374.3312 | 1.4010632
27 | 47 | 9 | 423 | 216.02625 | 1343.7625 | 0.56328166
28 | 53 | 9 | 477 | 166.081 | 995.6652 | 1.312338
29 | 60 | 9 | 540 | 110.92566 | 891.6857 | 0.26364383
30 | 70 | 9 | 630 | 117.78998 | 769.48975 | 1.1492673
31 | 16 | 9 | 144 | 272.28033 | 3616.8816 | 0.49877563
32 | 58 | 9 | 522 | 185.21194 | 986.7715 | 0.87296486
33 | 62 | 9 | 558 | 162.27156 | 1094.9476 | 1.2350211
34 | 48 | 9 | 432 | 185.67644 | 1592.98 | 1.0118079
```

35	46	9	414	227.2532	1368.7953	0.92758393
36	22	11	242	283.23026	4594.7017	0.47094786
37	52	9	468	172.25215	1181.5323	1.237589
38	49	9	441	144.47919	1139.4967	0.47356775
39	28	9	252	283.77878	2751.0374	0.6350357
40	51	11	561	250.27446	1245.5985	0.8673628
41	37	9	333	299.58527	4269.012	1.0417116
42	60	9	540	220.59433	2003.3375	0.8688995
43	54	10	540	272.94766	1717.6261	0.89578193
44	69	9	621	152.11028	951.35565	1.1064041
45	44	9	396	176.33061	781.39575	0.7057141
46	59	9	531	183.92982	910.3713	0.70241183
47	50	9	450	105.50018	709.34076	1.5004691
48	51	9	459	132.73166	765.07367	1.2716761
49	52	9	468	180.6069	1278.1661	1.2184353
50	47	10	470	185.66048	1087.8241	1.1128484
51	49	9	441	260.4043	1423.6848	0.6158534
52	51	9	459	178.07352	1157.6243	1.0225596
53	69	9	621	137.40973	709.6808	1.2341135
54	50	11	550	238.88904	1136.6309	1.3577094
55	43	9	387	128.5998	736.65204	0.9283251
56	67	9	603	169.27203	891.32556	1.457258
57	30	9	270	293.0383	1714.2323	0.52896357
58	87	9	783	117.41024	524.0207	1.057183
59	37	9	333	225.46613	1092.2501	0.7919304
60	54	9	486	142.14264	706.7783	1.5080202
61	67	9	603	145.87569	590.3016	1.0237952
62	41	10	410	87.59235	492.93777	1.3329939
63	49	9	441	167.9097	858.06146	1.045213
64	37	9	333	284.8295	2080.4265	0.6546314
65	60	9	540	183.70737	811.96173	1.1731919
66	62	9	558	171.89948	854.1047	1.5769937
67	59	10	590	157.2245	866.7605	2.0961096
68	16	12	192	295.86438	3553.9001	0.27684236
69	46	9	414	178.85582	780.52356	1.409748
70	48	9	432	217.12003	1267.6349	1.3374622
71	12	9	108	265.39438	3867.0854	0.31799522
72	35	9	315	274.22595	1488.8928	0.845789
73	52	9	468	177.57858	980.40906	1.8483915
74	49	9	441	180.20918	929.6857	1.2710242
75	26	9	234	295.19168	3006.4026	0.8456667
76	11	10	110	289.2919	4267.385	1.0999613
77	52	9	468	222.00096	1016.5293	1.0660915
78	58	9	522	182.44415	1148.1584	1.3128413

```
87   79 | 55 | 9 | 495 | 230.19388 | 1060.4109 | 0.7344301
88   80 | 56 | 9 | 504 | 142.15656 | 788.9491 | 1.455978
89   81 | 21 | 9 | 189 | 293.65256 | 1986.8163 | 1.1027952
90   82 | 28 | 9 | 252 | 295.1973 | 3538.7993 | 1.1947818
91   83 | 49 | 9 | 441 | 207.34674 | 1306.7075 | 1.8227099
92   84 | 68 | 9 | 612 | 234.18875 | 1231.8767 | 1.3995994
93   85 | 64 | 9 | 576 | 108.81802 | 557.7322 | 1.7873535
94   86 | 53 | 9 | 477 | 136.38461 | 823.5026 | 0.7563948
95   87 | 53 | 10 | 530 | 289.3597 | 1738.789 | 1.3140296
96   88 | 40 | 9 | 360 | 202.9854 | 1390.6 | 1.4368641
97   89 | 5 | 10 | 50 | 254.7936 | 5097.2886 | 0.1426294
98   90 | 60 | 9 | 540 | 133.96584 | 631.4467 | 1.49286
99   91 | 41 | 9 | 369 | 197.94957 | 1342.2994 | 1.57238
100  92 | 35 | 9 | 315 | 297.37024 | 2950.0503 | 1.0586798
101  93 | 54 | 9 | 486 | 174.19911 | 1336.0432 | 1.4973605
102  94 | 55 | 10 | 550 | 146.30281 | 599.1107 | 1.896195
103  95 | 74 | 9 | 666 | 116.7339 | 625.80994 | 2.309831
104  96 | 10 | 11 | 110 | 256.37842 | 2182.173 | 0.3528584
105  97 | 6 | 9 | 54 | 227.16595 | 6261.4956 | 0.1734241
106  98 | 56 | 9 | 504 | 174.2462 | 854.42664 | 1.9160883
107  99 | 1 | 12 | 12 | 0 | 10362.077 | 3.40E+38
108  100 | 40 | 9 | 360 | 118.39034 | 1077.948 | 1.399147
```

E.1.6 Opérateurs génétiques intelligents : 100 essais

```
result_yeast_02022010010432.txt         Temps total d'execution : 2487 sec
Coverage: 0.4346292
Overlapping: 0.42615795
Genes: 0.61581135
Conditions: 1.0
Bicluster | Rows | Columns | Volume | Residue | Row Variance | Fitness
Moyenne | 39.33 | 9.16 | 357.78 | 187.7156 | 1724.7152 | 0.80532247
Ecart type | 14.888959 | 0.4841489 | 132.82272 | 59.983612 | 1250.8945 | 0.4395535
1 | 35 | 9 | 315 | 199.21443 | 3533.3909 | 0.17606865
2 | 51 | 9 | 459 | 218.89445 | 1333.722 | 0.18664633
3 | 62 | 9 | 558 | 119.23418 | 1034.74 | 0.24071072
4 | 50 | 9 | 450 | 190.18741 | 1677.1462 | 0.5208409
5 | 63 | 9 | 567 | 171.02626 | 1096.6294 | 0.15640007
6 | 50 | 9 | 450 | 199.97733 | 1703.3057 | 0.29035807
7 | 38 | 9 | 342 | 136.57922 | 1056.2999 | 0.50458395
8 | 37 | 9 | 333 | 279.21332 | 2440.5083 | 0.18659128
9 | 51 | 9 | 459 | 196.02507 | 1962.3195 | 0.37444857
10 | 30 | 10 | 300 | 173.32408 | 1315.166 | 0.33441824
11 | 36 | 9 | 324 | 185.86726 | 1363.6407 | 0.19222736
12 | 62 | 9 | 558 | 158.73042 | 869.50885 | 0.56358683
13 | 55 | 9 | 495 | 135.23557 | 920.7964 | 0.47396946
14 | 56 | 9 | 504 | 243.4119 | 1798.8809 | 0.4000988
15 | 20 | 10 | 200 | 277.5813 | 3729.757 | 0.1311878
16 | 34 | 9 | 306 | 225.64816 | 2194.3118 | 0.3691138
17 | 60 | 9 | 540 | 114.42862 | 874.0688 | 0.70130897
18 | 54 | 9 | 486 | 168.60153 | 907.6006 | 0.5281
19 | 40 | 9 | 360 | 205.53592 | 1788.0927 | 0.52784544
20 | 38 | 9 | 342 | 188.92729 | 1560.233 | 0.52046424
21 | 49 | 9 | 441 | 180.94249 | 1502.2633 | 1.0068188
22 | 35 | 9 | 315 | 171.0822 | 1302.417 | 0.6018039
23 | 67 | 9 | 603 | 92.17875 | 702.3394 | 0.97773755
24 | 35 | 9 | 315 | 93.15626 | 1140.7407 | 0.28585014
25 | 42 | 9 | 378 | 181.17438 | 1627.1251 | 0.6147726
26 | 56 | 9 | 504 | 145.27193 | 988.9876 | 0.76434207
27 | 45 | 9 | 405 | 273.6082 | 1947.353 | 0.49823833
28 | 65 | 9 | 585 | 144.31885 | 931.35 | 0.8404475
29 | 40 | 10 | 400 | 212.55365 | 1372.4521 | 0.48505583
30 | 13 | 9 | 117 | 246.1499 | 4079.0103 | 0.14859301
31 | 47 | 9 | 423 | 202.06134 | 1268.9465 | 0.57473123
32 | 49 | 9 | 441 | 133.34666 | 755.1643 | 0.5750513
33 | 19 | 11 | 209 | 284.65366 | 4770.651 | 0.6708433
34 | 53 | 9 | 477 | 89.715706 | 832.27374 | 0.2625416
```

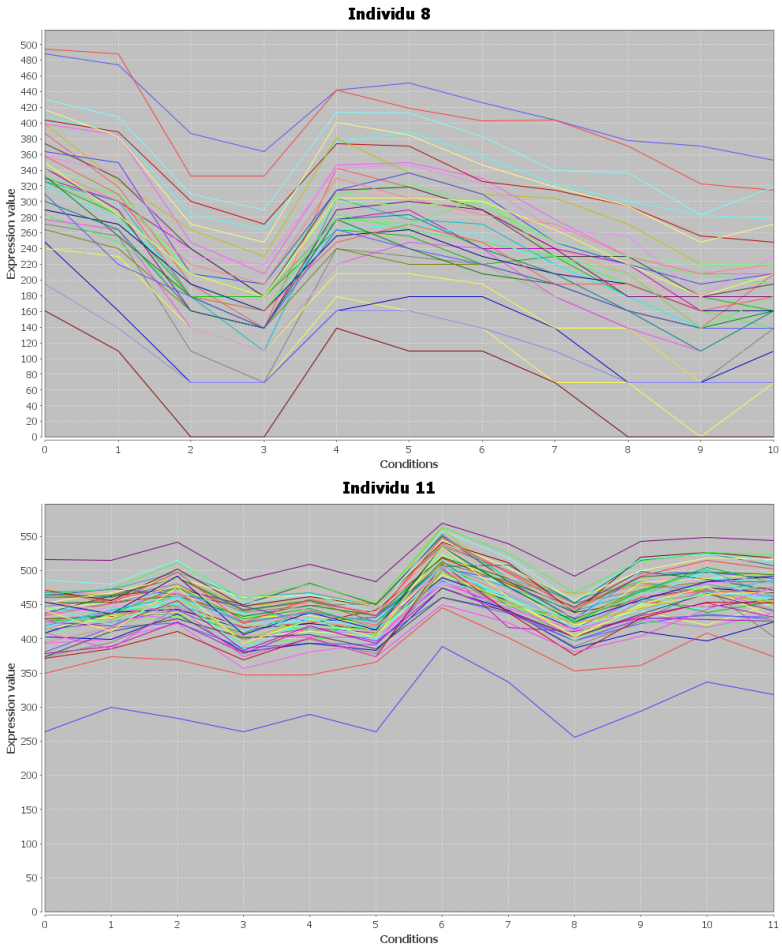
43	35	21	9	189	284.96347	3383.551	0.63266486
44	36	23	9	207	277.1047	2350.2664	0.39111644
45	37	13	9	117	298.76144	3985.1885	0.25258216
46	38	59	9	531	81.06638	619.47687	1.2337203
47	39	48	9	432	92.655334	655.18933	1.0467858
48	40	30	9	270	154.54128	967.66626	1.0725174
49	41	51	9	459	209.98198	1378.5308	0.7084198
50	42	27	9	243	176.28224	1245.9785	1.0470338
51	43	12	9	108	254.55862	3968.8037	0.2884465
52	44	24	9	216	86.56579	724.9656	0.80183506
53	45	35	9	315	195.89792	1825.5399	1.0503178
54	46	40	9	360	219.83882	1698.7915	0.7845486
55	47	51	9	459	170.85219	1064.65	1.1176299
56	48	47	9	423	204.96863	1446.0645	0.7937386
57	49	43	9	387	227.73125	1267.25	1.0531827
58	50	12	12	144	267.8626	2276.798	0.403036
59	51	27	9	243	292.61246	3437.8684	0.47596145
60	52	39	9	351	148.36626	1109.4366	0.96256584
61	53	9	10	90	244.98418	2483.4983	0.1942539
62	54	51	9	459	146.05621	828.1787	1.137351
63	55	41	9	369	116.38776	809.41974	1.0746888
64	56	20	9	180	299.7032	4107.146	0.4592204
65	57	54	9	486	120.87925	681.26447	1.1712754
66	58	16	9	144	273.18295	2738.8906	0.52589947
67	59	40	9	360	142.79506	913.9315	0.3628683
68	60	49	10	490	104.11296	496.46112	0.9209902
69	61	73	9	657	138.85832	718.3281	1.2006825
70	62	39	9	351	224.37898	1550.9755	0.9736426
71	63	3	9	27	105.54458	7680.559	0.07901953
72	64	35	9	315	220.38206	1571.2867	0.81966925
73	65	27	9	243	279.6668	2468.2383	0.90654916
74	66	41	9	369	196.26637	1304.947	0.79216063
75	67	50	9	450	186.86818	1609.8146	1.6575501
76	68	60	9	540	155.07124	1060.1213	1.2299957
77	69	35	9	315	184.05823	1467.9105	1.0336342
78	70	34	9	306	261.09467	1817.7751	1.1151412
79	71	52	10	520	157.64774	708.5456	1.1905048
80	72	57	9	513	121.93681	904.67395	1.5369327
81	73	46	9	414	146.02837	544.4295	0.54097307
82	74	41	9	369	98.09455	627.42664	1.2399766
83	75	35	9	315	197.30127	1008.9126	0.7824749
84	76	40	9	360	241.84695	2314.7354	0.76442844
85	77	48	9	432	186.421	1019.5632	1.3033116
86	78	31	9	279	192.62463	1392.7866	1.0053977

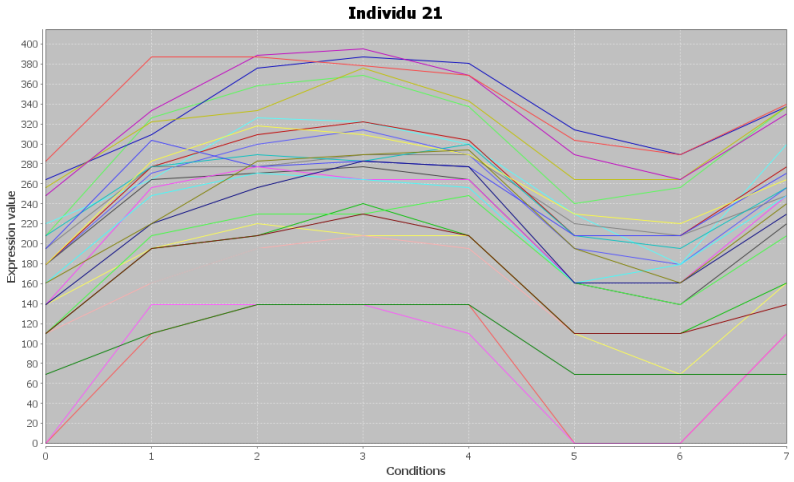
```
 87   79 | 51 | 9 | 459 | 97.119865 | 546.3489 | 1.4637173
 88   80 | 29 | 10 | 290 | 267.46875 | 1599.9573 | 0.5738731
 89   81 | 45 | 9 | 405 | 172.94368 | 1054.866 | 1.7381535
 90   82 | 61 | 9 | 549 | 135.14836 | 775.34064 | 1.6118345
 91   83 | 33 | 9 | 297 | 117.66848 | 634.57916 | 1.6093452
 92   84 | 25 | 9 | 225 | 257.60715 | 1540.9993 | 0.95581067
 93   85 | 6 | 11 | 66 | 203.84436 | 5560.4253 | 0.6613637
 94   86 | 36 | 9 | 324 | 270.62134 | 2270.7068 | 1.3719765
 95   87 | 40 | 9 | 360 | 139.27835 | 701.1784 | 1.2851895
 96   88 | 30 | 9 | 270 | 130.3507 | 980.95056 | 1.0764016
 97   89 | 49 | 9 | 441 | 80.31505 | 822.55615 | 1.0554912
 98   90 | 23 | 10 | 230 | 207.82657 | 2020.1594 | 0.9164181
 99   91 | 5 | 9 | 45 | 243.47557 | 5363.021 | 0.049502186
100   92 | 46 | 9 | 414 | 190.14064 | 1104.9122 | 1.3061111
101   93 | 28 | 9 | 252 | 265.10843 | 4148.6465 | 1.2218008
102   94 | 57 | 9 | 513 | 84.45634 | 414.62674 | 1.5782961
103   95 | 30 | 9 | 270 | 241.16933 | 3110.8745 | 1.3532631
104   96 | 29 | 9 | 261 | 240.2419 | 2270.5112 | 1.1499652
105   97 | 34 | 9 | 306 | 263.49347 | 1459.3116 | 0.84790576
106   98 | 26 | 9 | 234 | 222.36694 | 1637.6473 | 1.1923602
107   99 | 41 | 9 | 369 | 183.96143 | 1155.7457 | 1.8104849
108   100 | 43 | 10 | 430 | 132.28847 | 681.1506 | 1.8824935
```

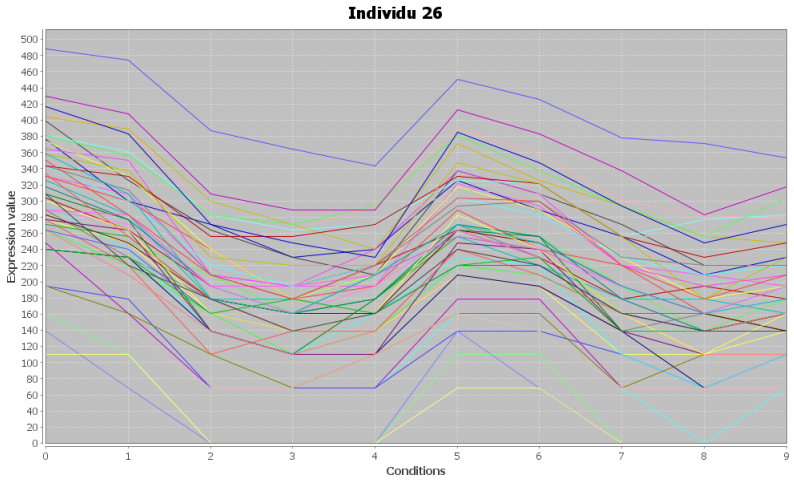
Annexe F

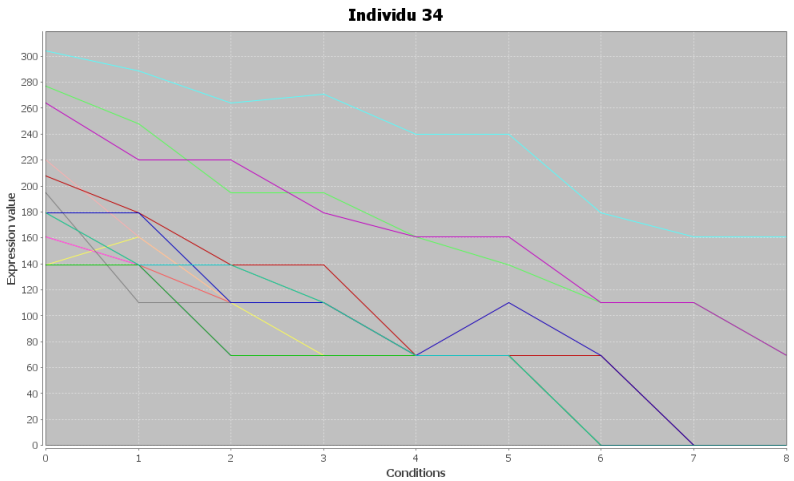
Résultats individuels SMOB et améliorations

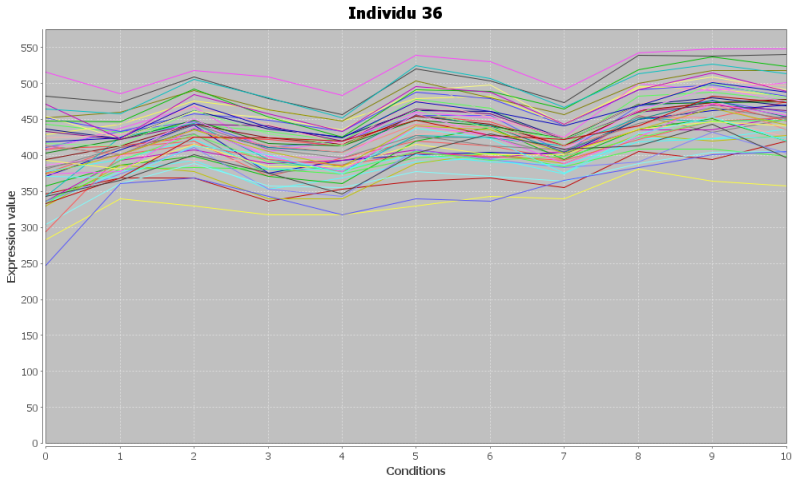
F.1 Yeast dataset

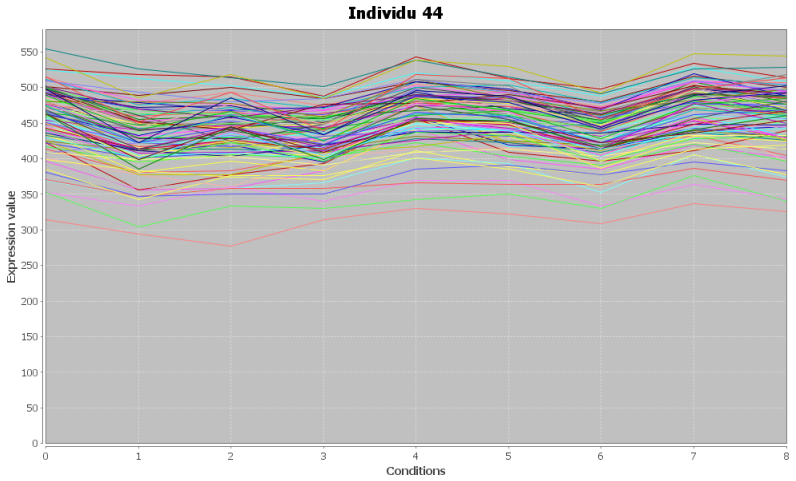


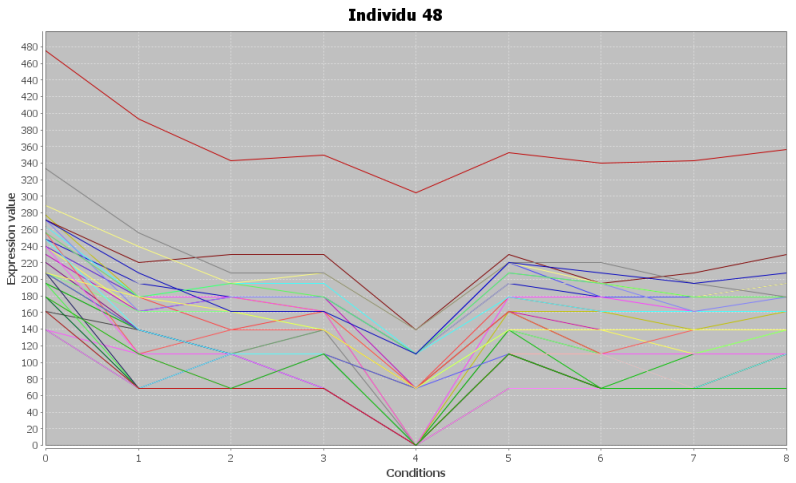

Individu 21
Expression value
Conditions

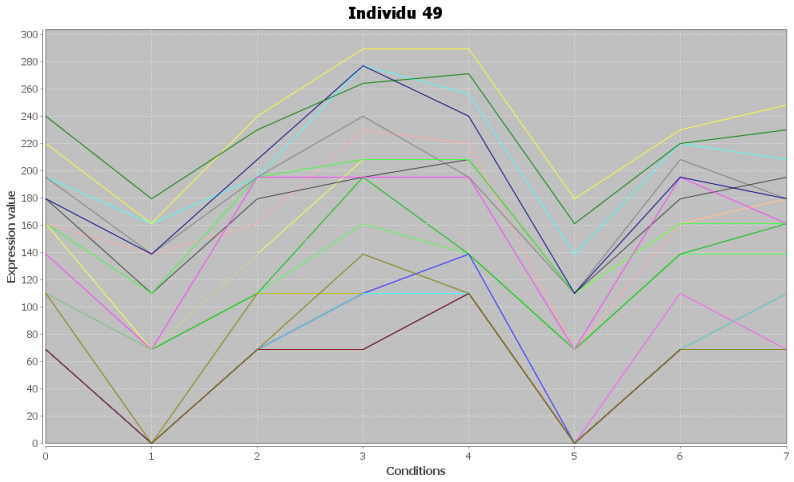

Individu 26
Expression value
Conditions

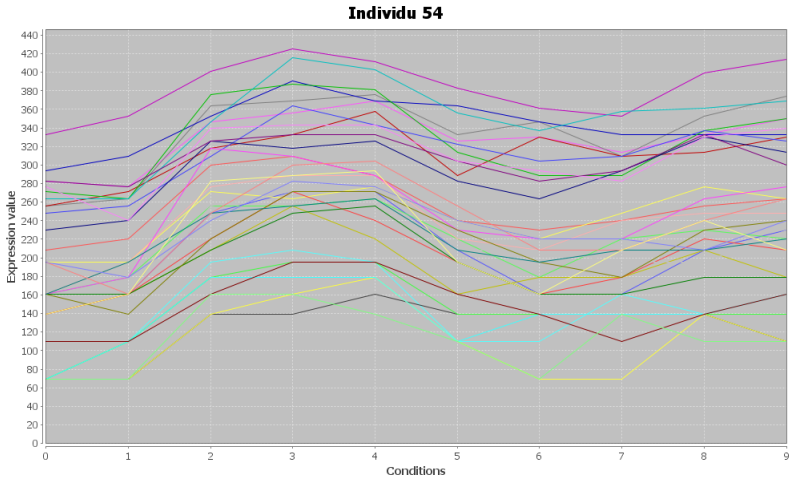

Individu 34
Expression value
Conditions

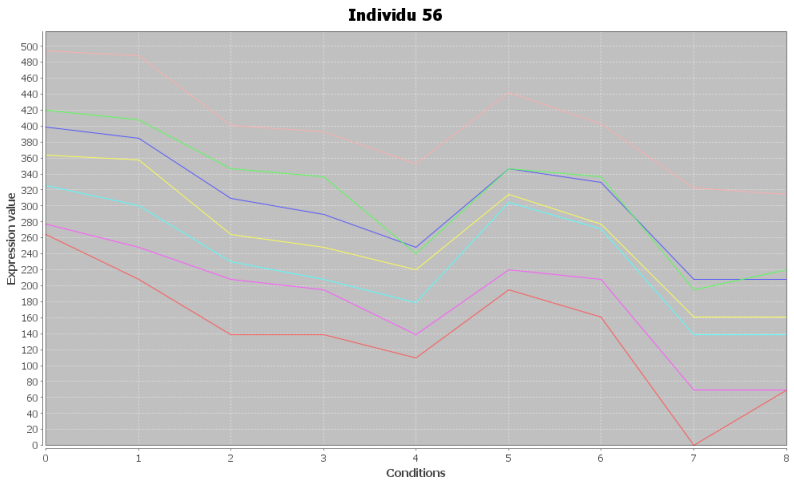

Individu 36
Expression value
Conditions

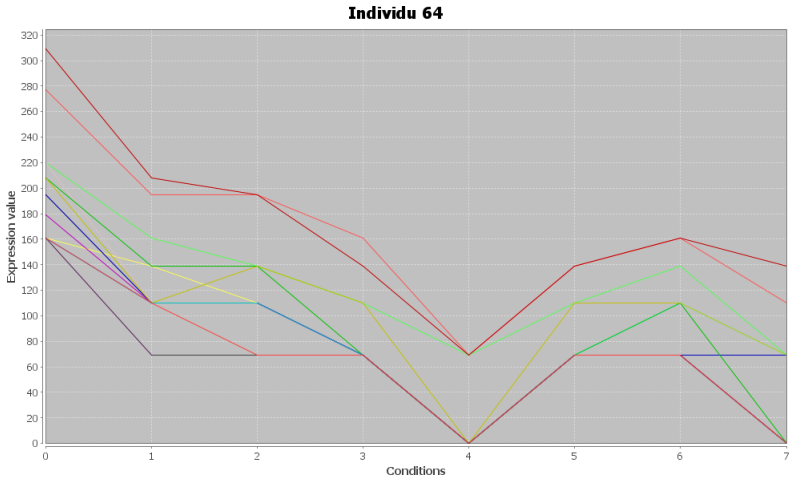

Individu 44
Expression value
Conditions

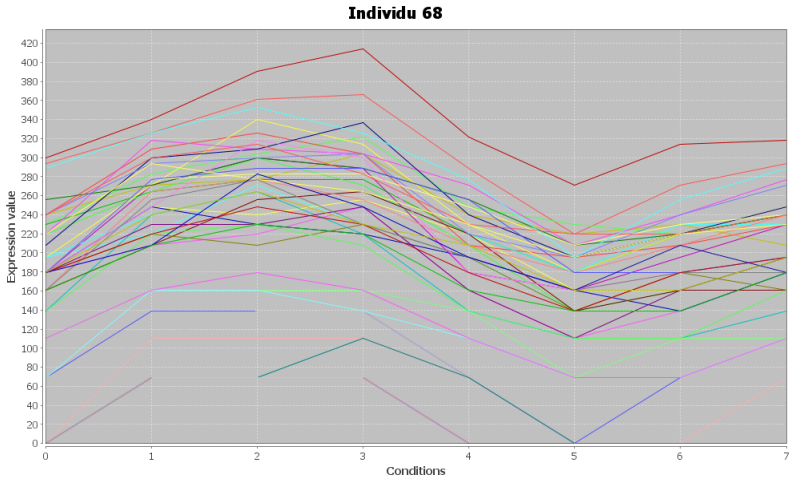

Individu 48
Expression value
Conditions

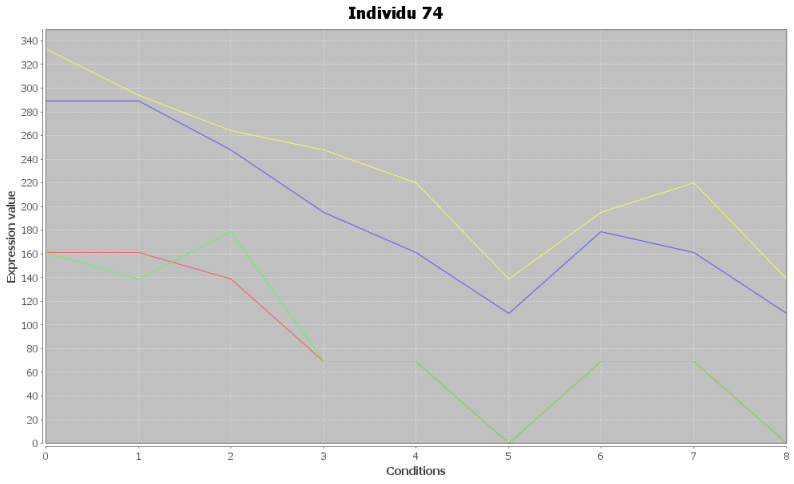

Individu 49
Expression value
Conditions

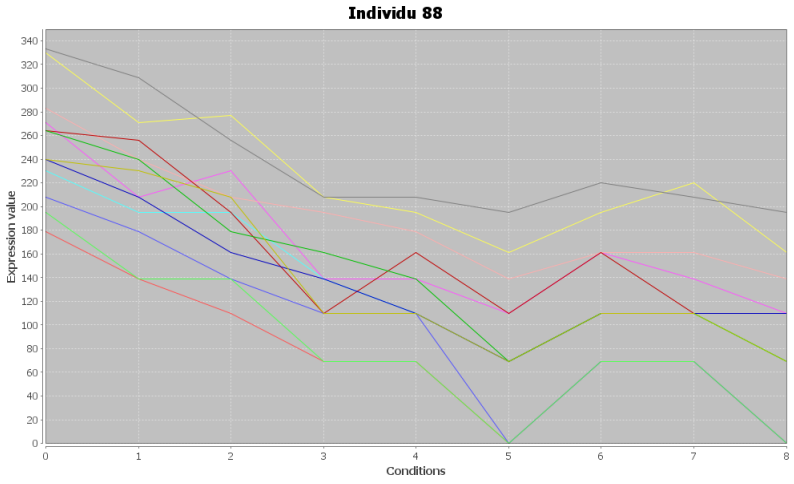

Individu 54
Expression value
Conditions

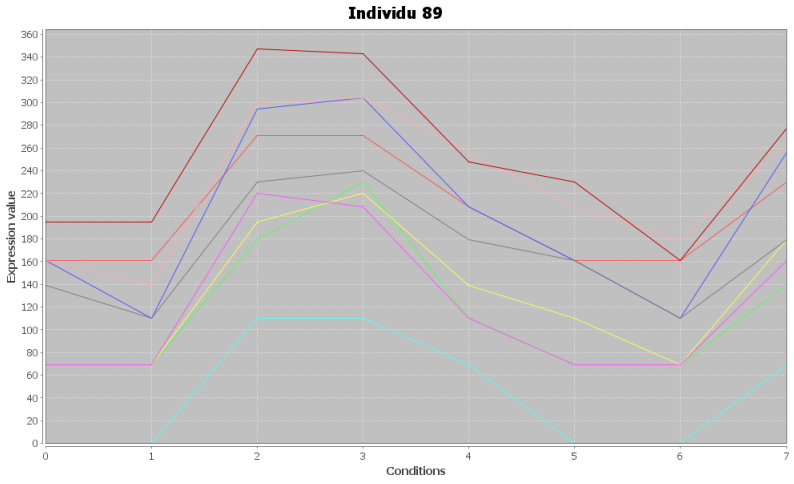

Individu 56
Expression value
Conditions

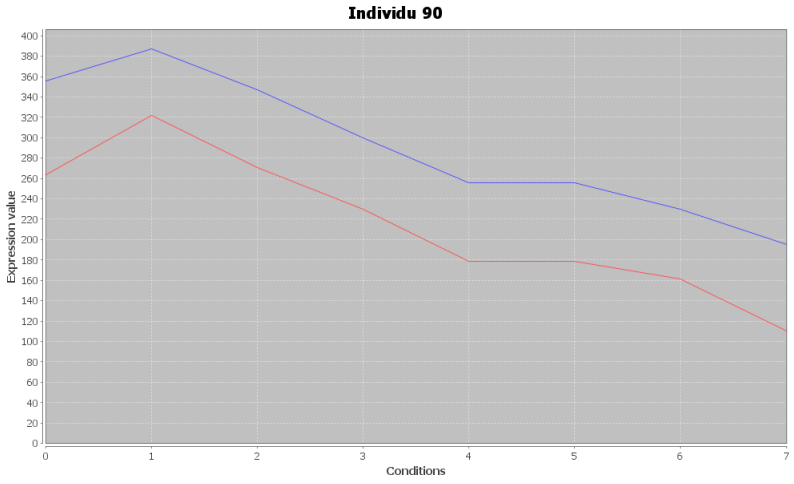

Individu 64
Expression value
Conditions

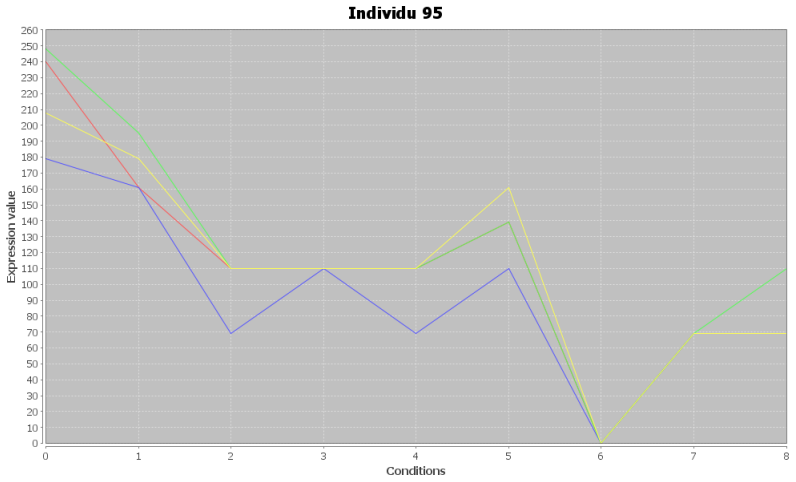

Individu 68
Expression value
Conditions

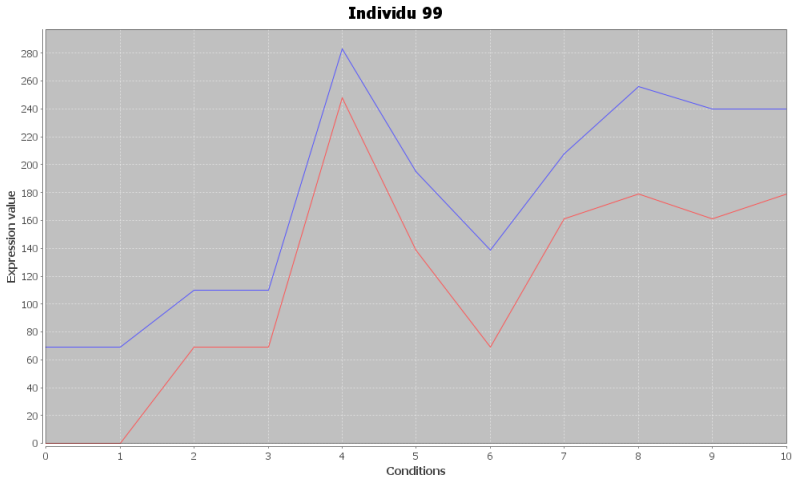

Individu 74
Expression value
Conditions

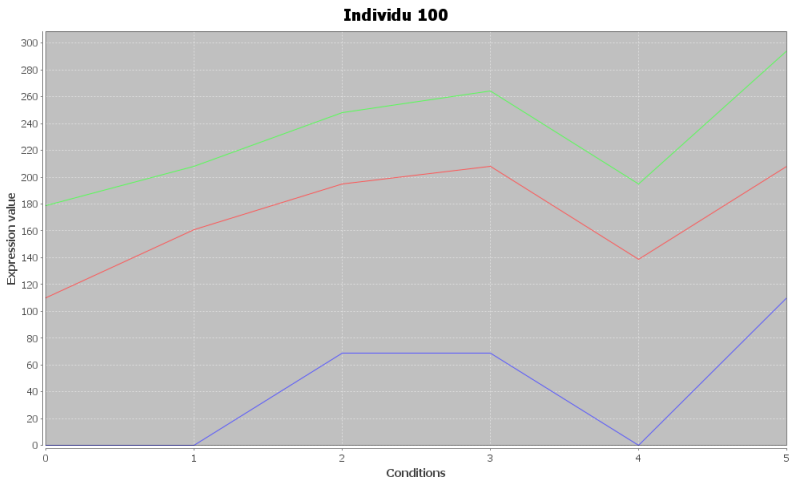

Individu 88
Expression value
Conditions


Individu 89
Expression value
Conditions

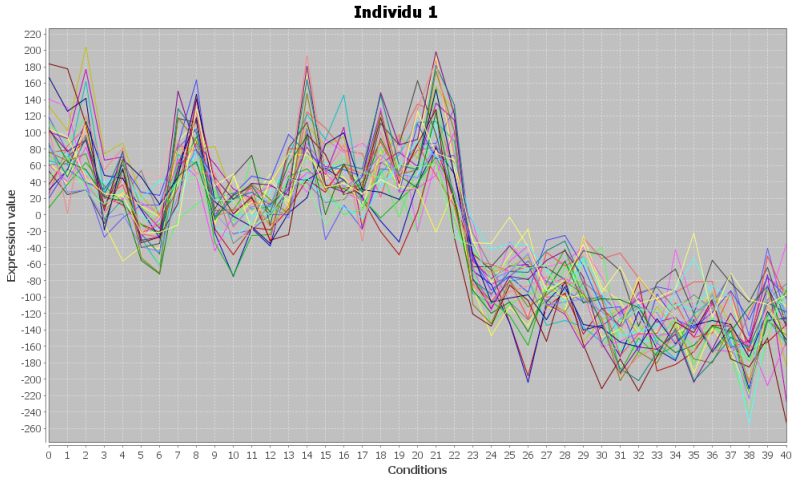

Individu 90
Expression value
Conditions

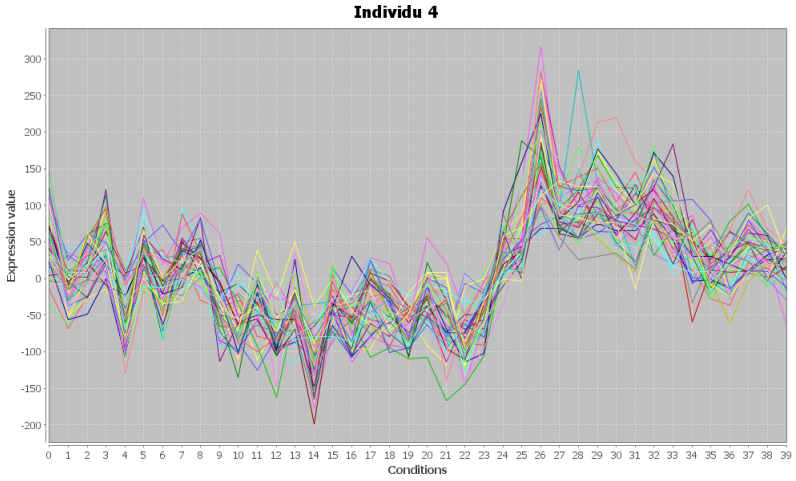

Individu 95
Expression value
Conditions

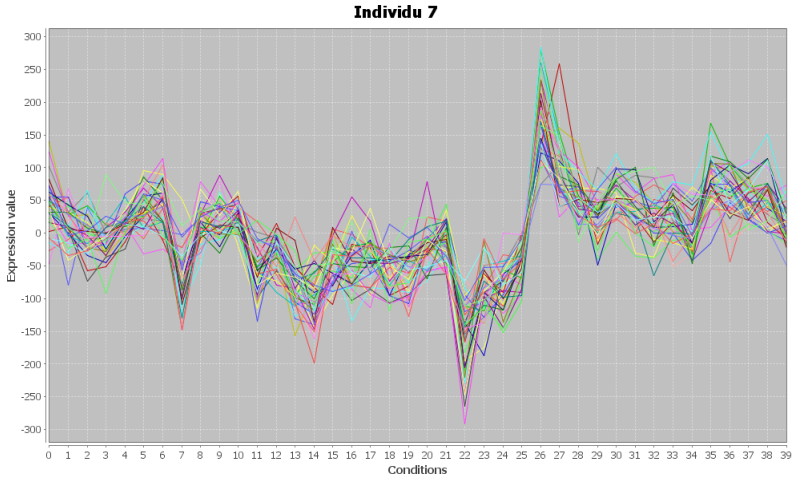

Individu 99
Expression value
Conditions

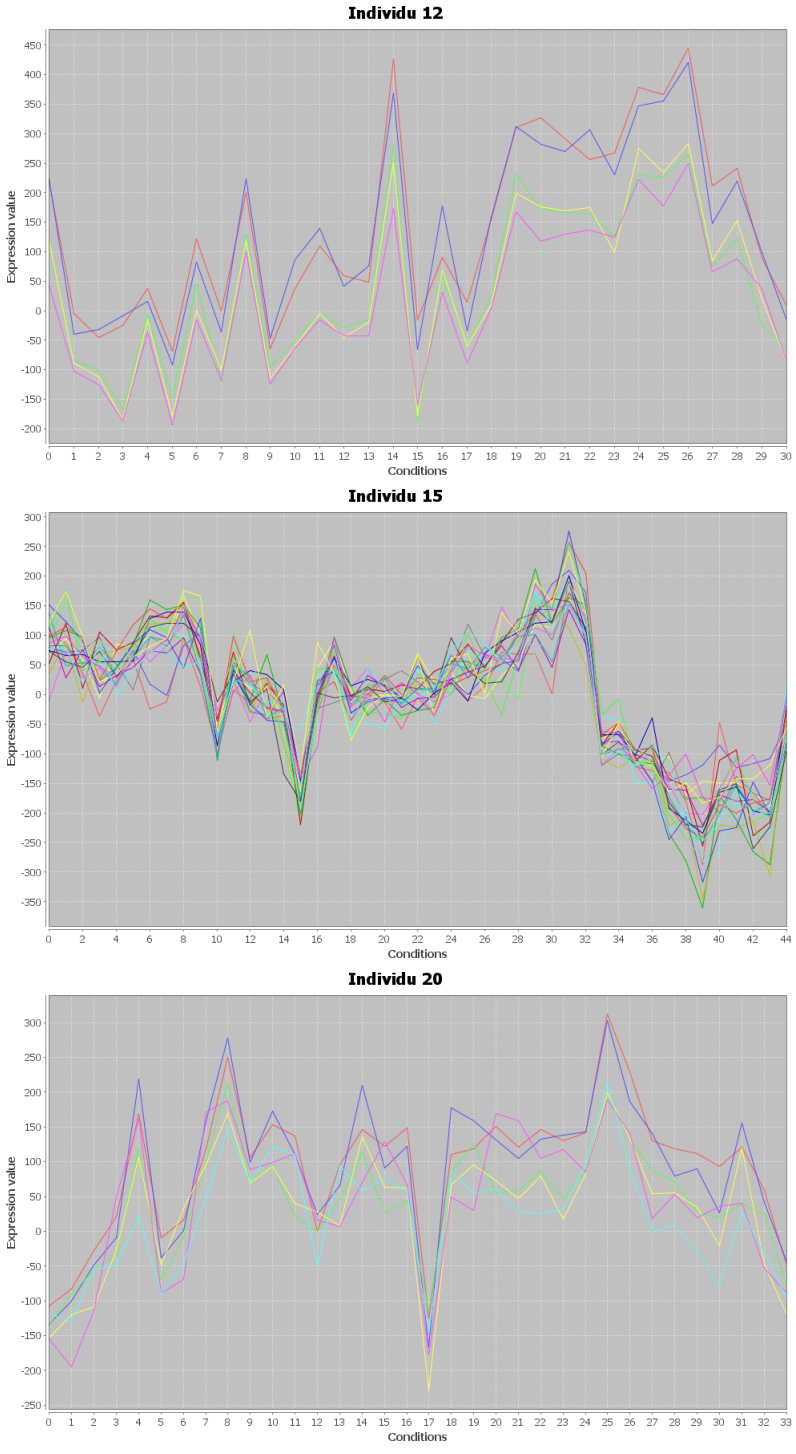

Individu 100
Expression value
Conditions

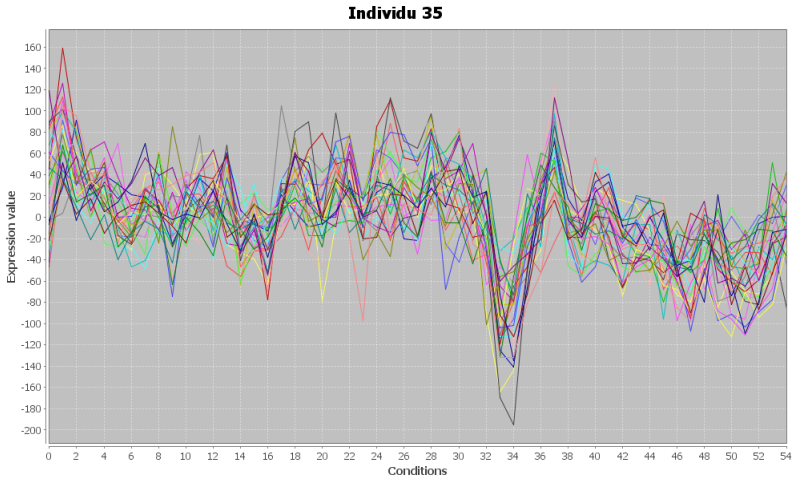
F.2 Human Lymphoma dataset

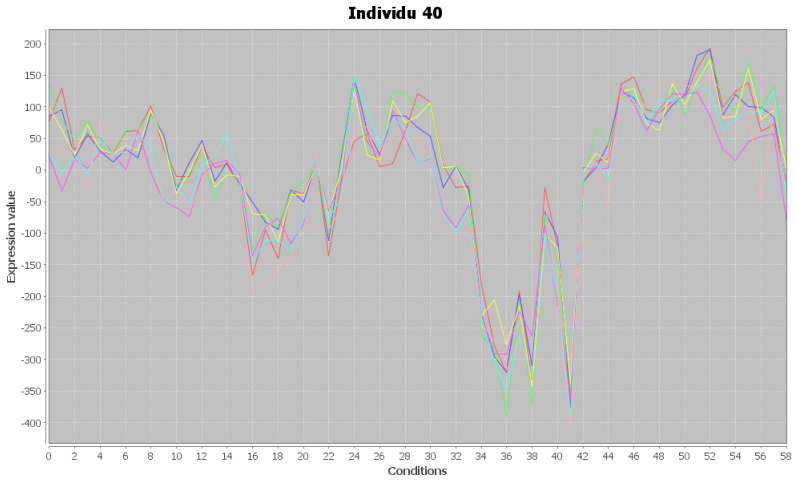


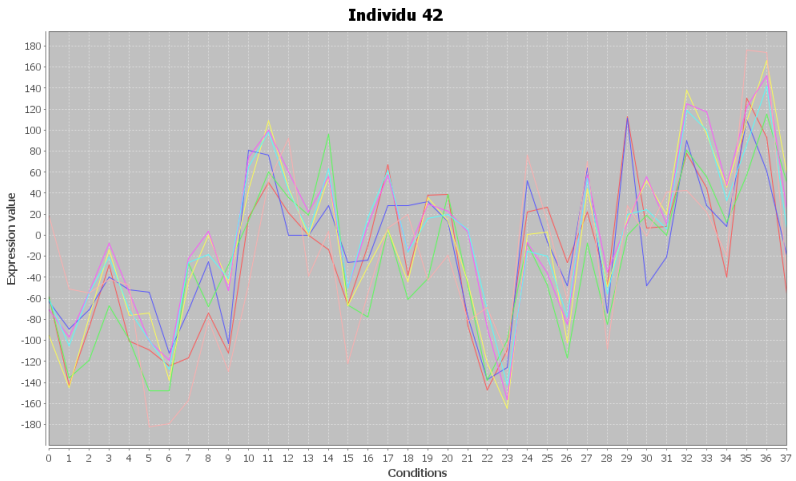


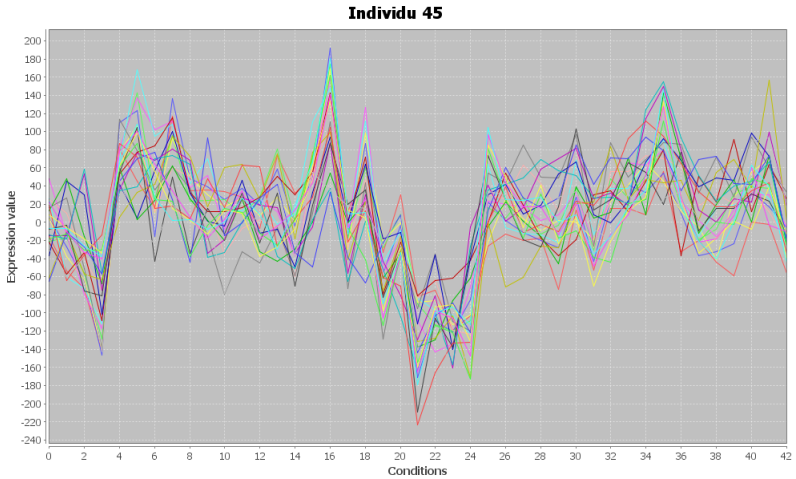


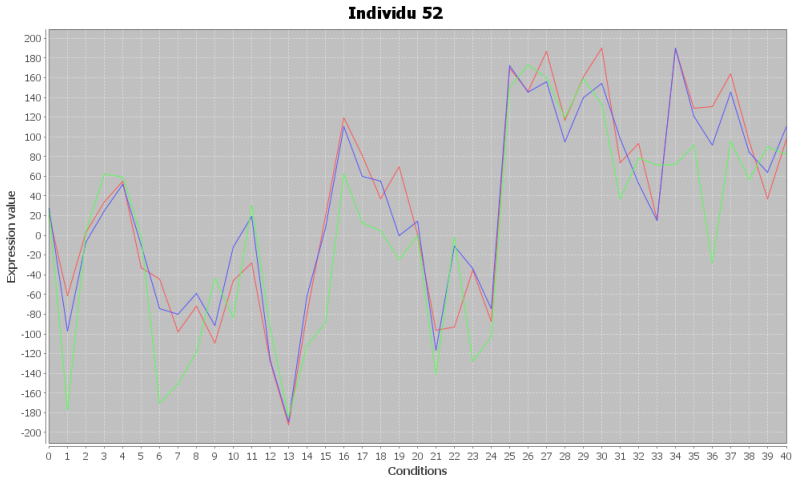

Individu 12
Expression value
Conditions
Individu 15
Expression value
Conditions
Individu 20
Expression value
Conditions

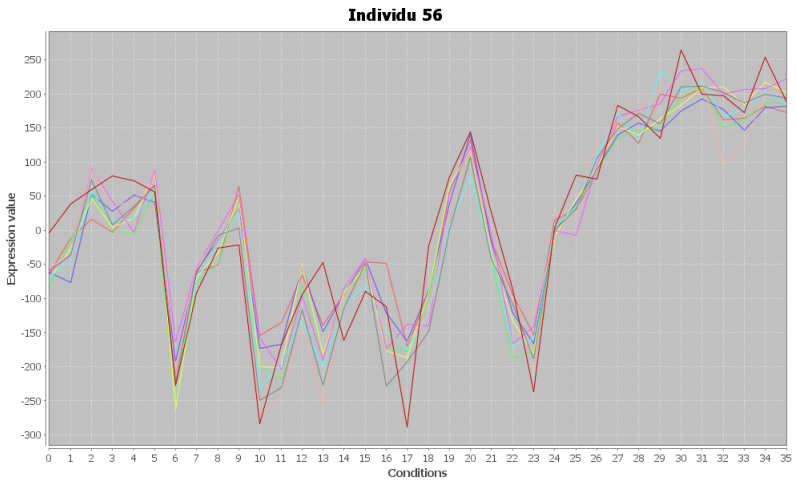

Individu 35
Expression value
Conditions

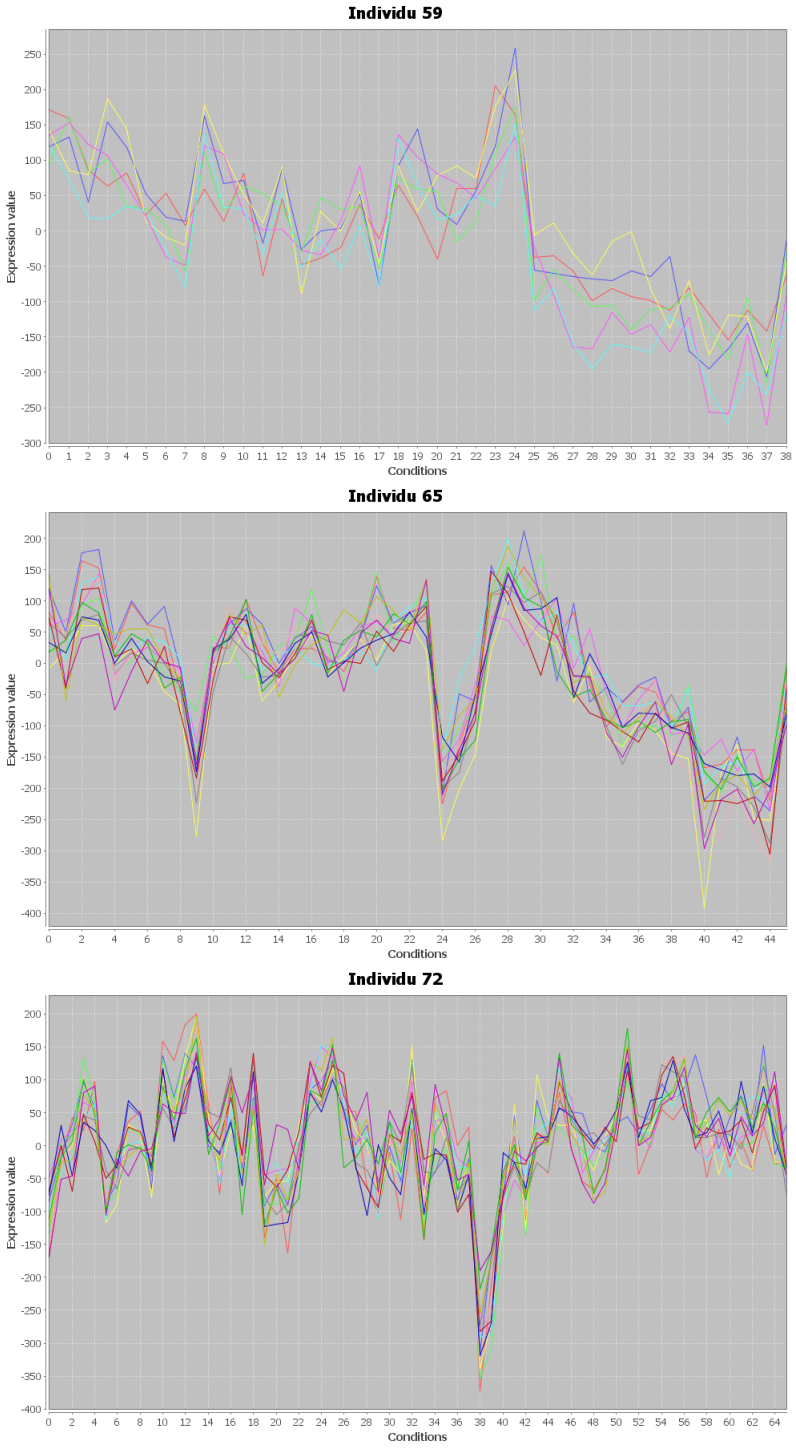

Individu 40
Expression value
Conditions

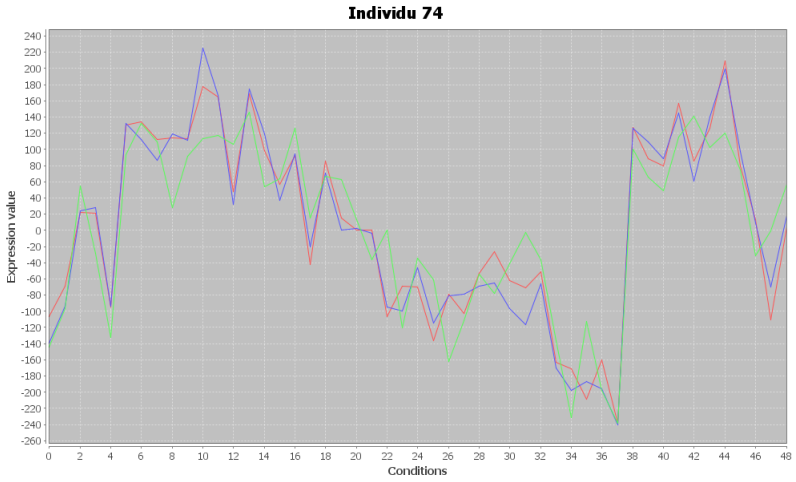

Individu 42
Expression value
Conditions

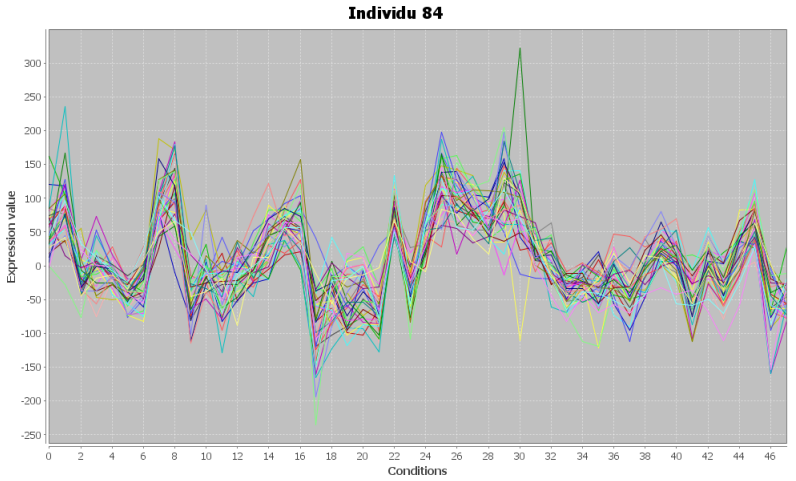

Individu 45
Expression value
Conditions

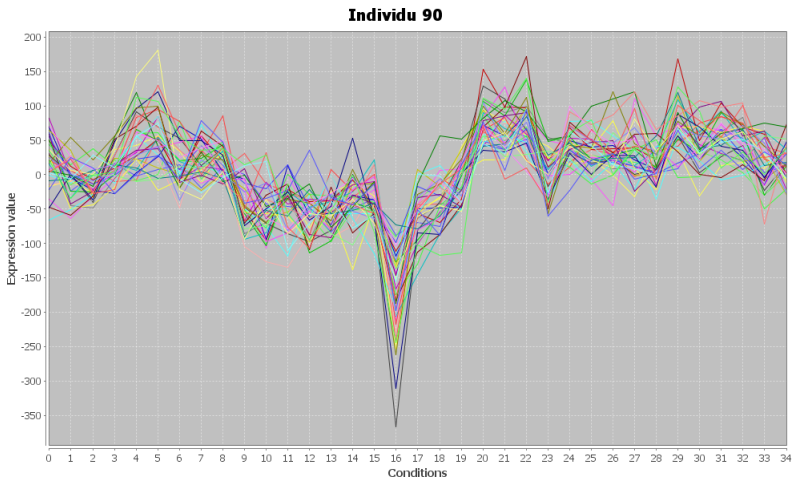

Individu 52
Expression value
Conditions

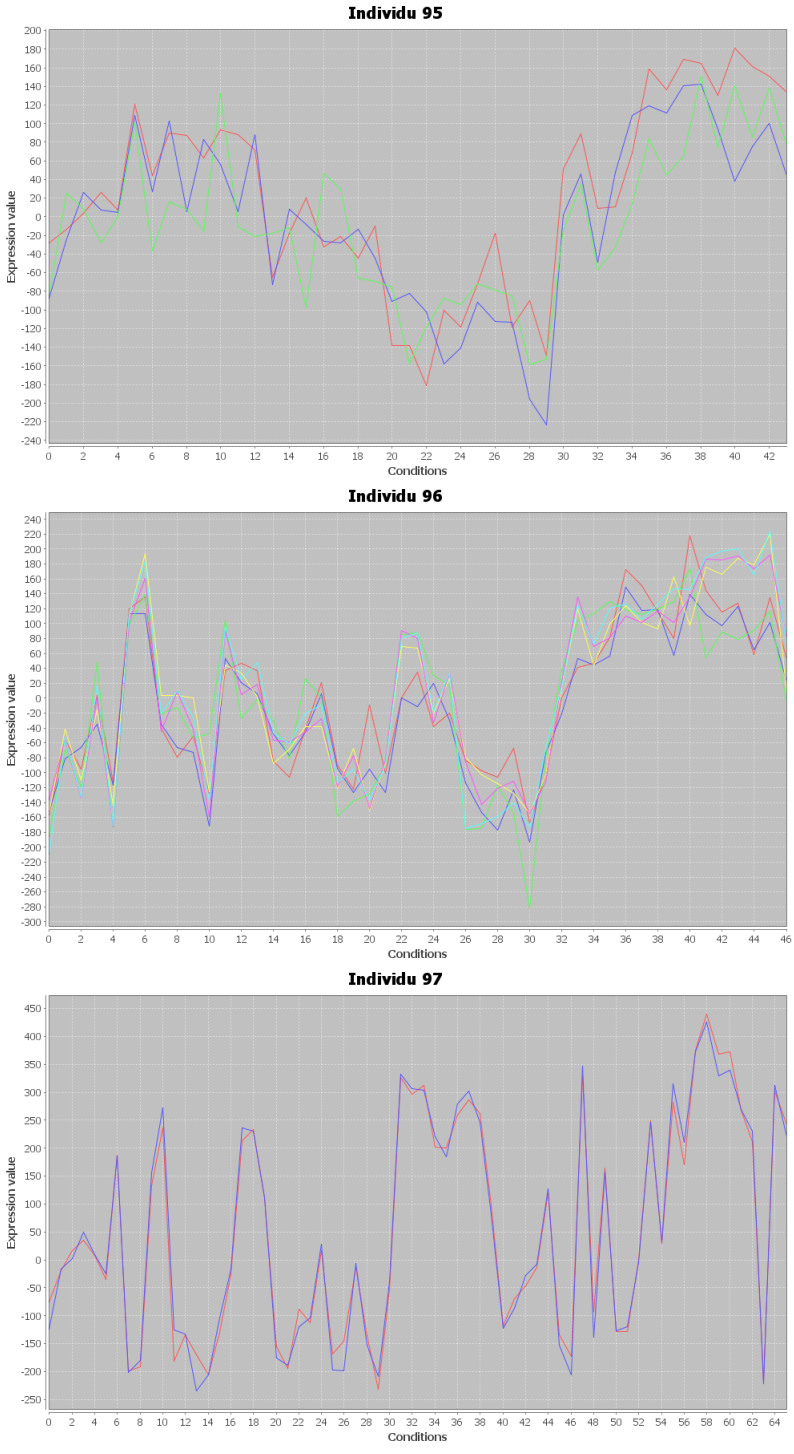

Individu 56
Expression value
Conditions


Individu 59
Expression value
Conditions
Individu 65
Expression value
Conditions
Individu 72
Expression value
Conditions

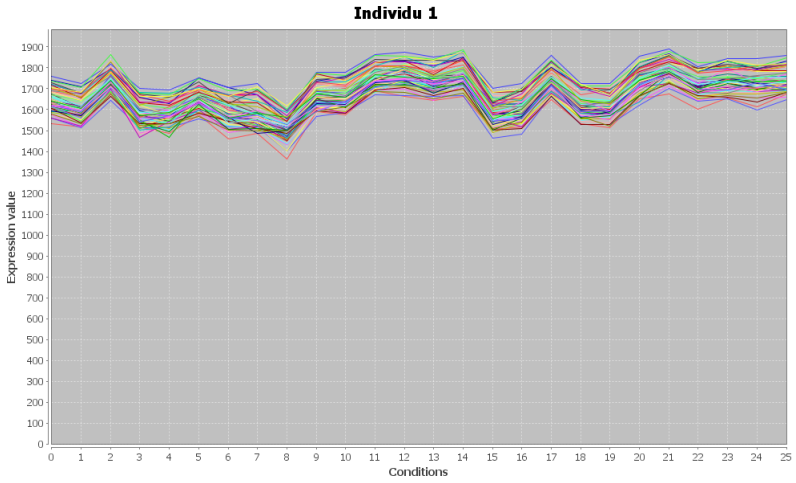

Individu 74
Expression value
Conditions

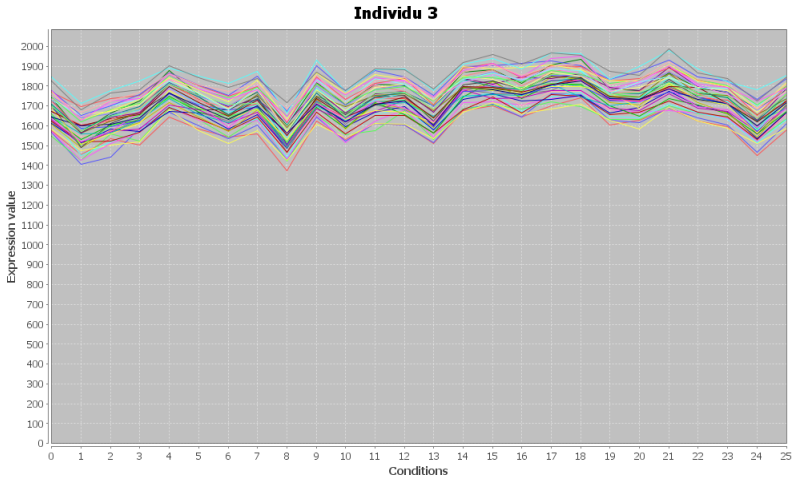

Individu 84
Expression value
Conditions

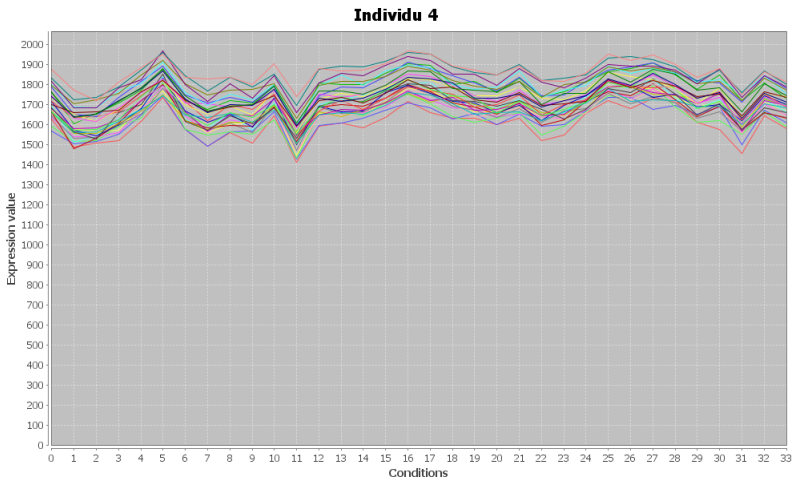

Individu 90
Expression value
Conditions

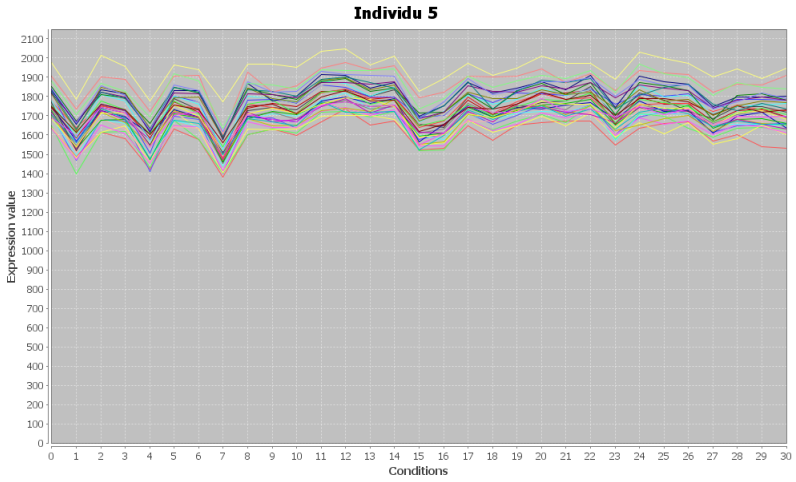

Individu 95
Expression value
Conditions
Individu 96
Expression value
Conditions
Individu 97
Expression value
Conditions

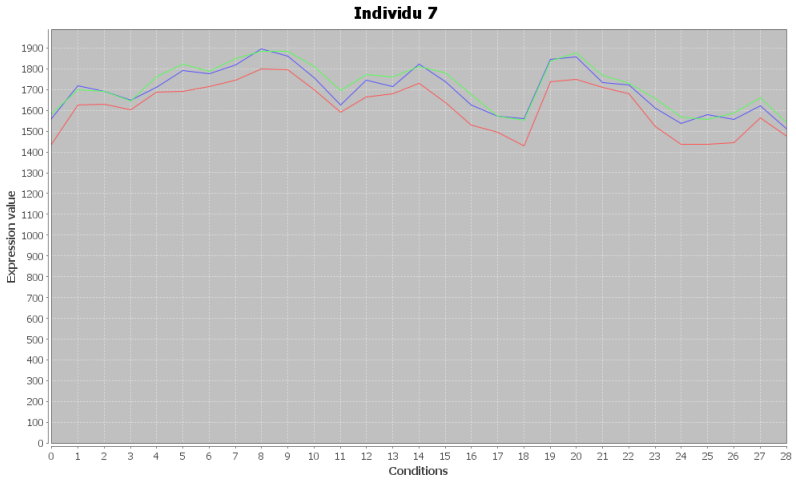
F.3 Colon Cancer dataset

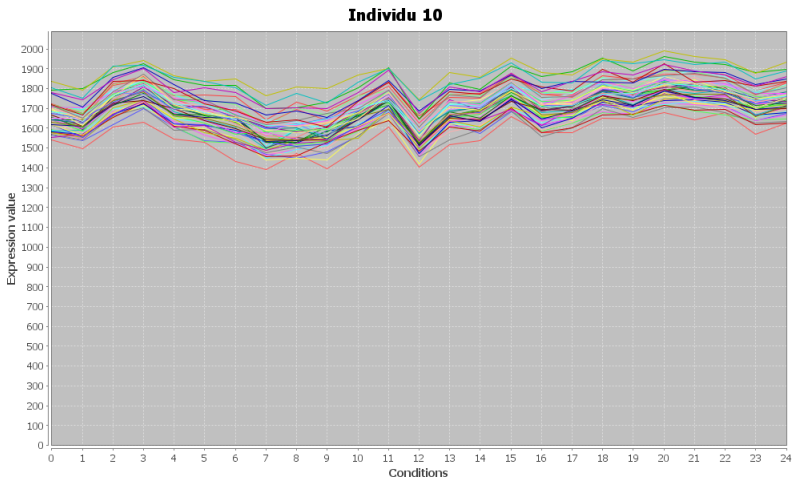


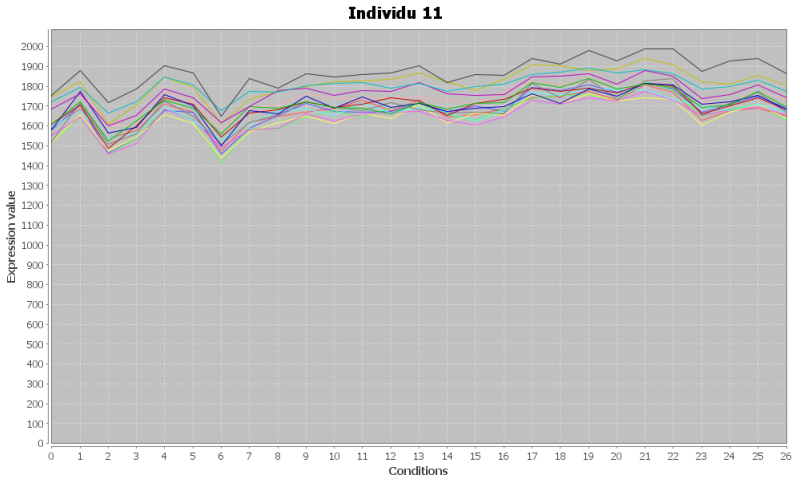


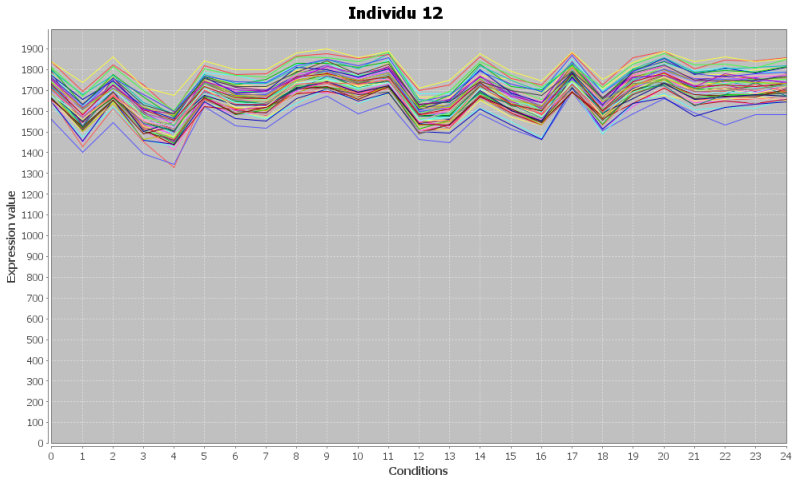


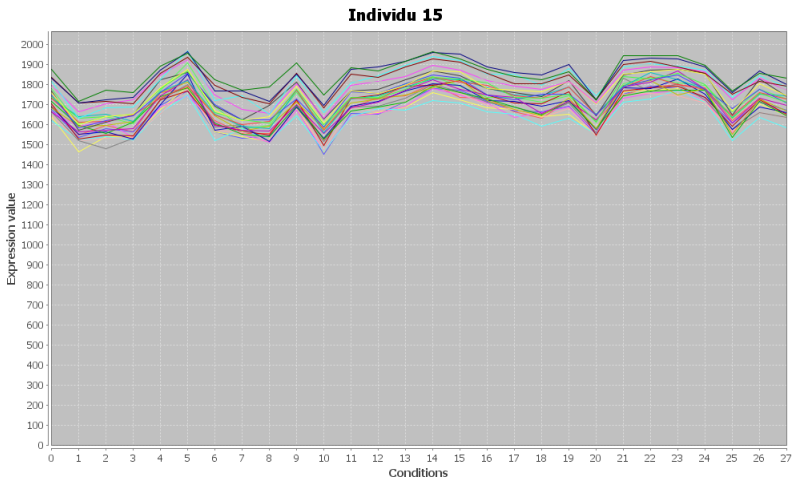

Individu 5
Expression value
Conditions

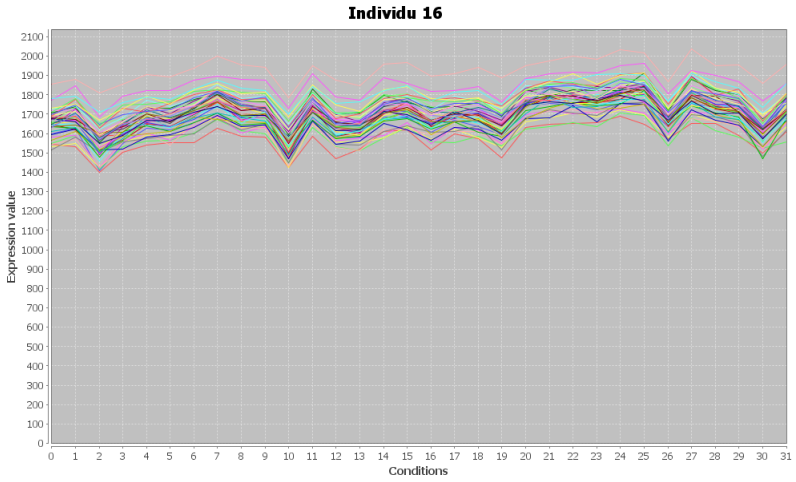

Individu 7
Expression value
Conditions

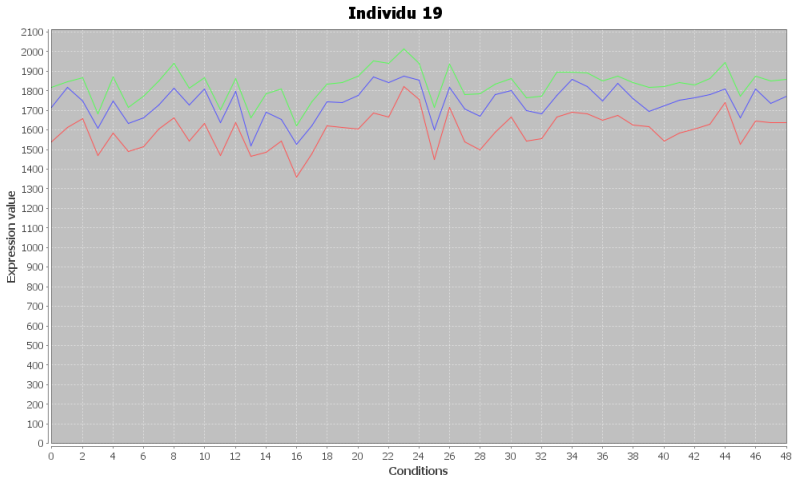

Individu 10
Expression value
Conditions

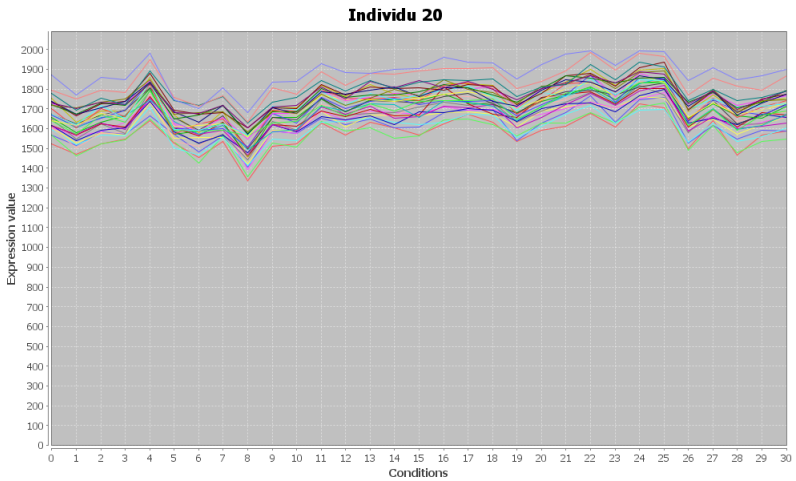

Individu 11
Expression value
Conditions

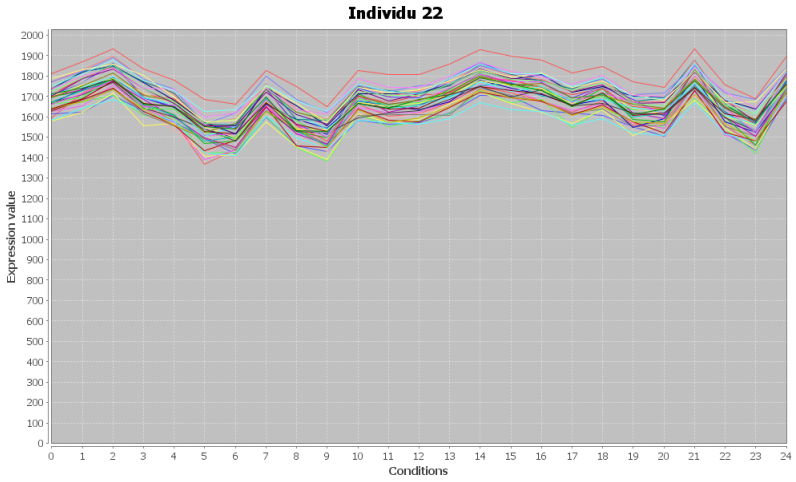

Individu 12
Expression value
Conditions

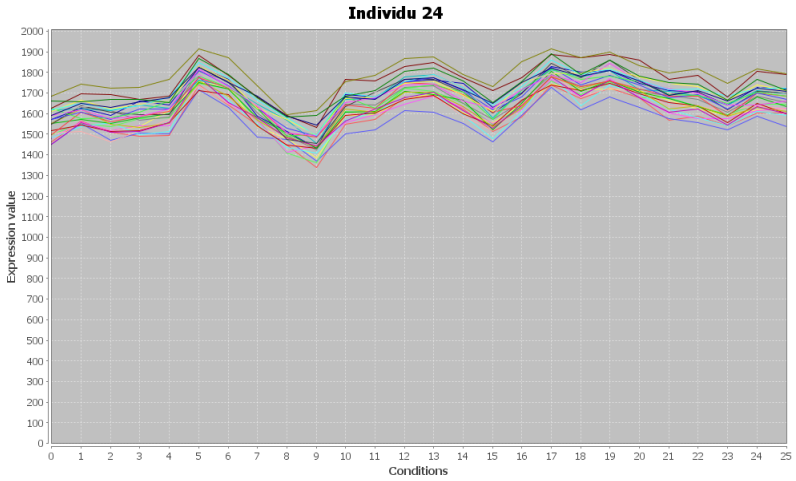

Individu 15
Expression value
Conditions

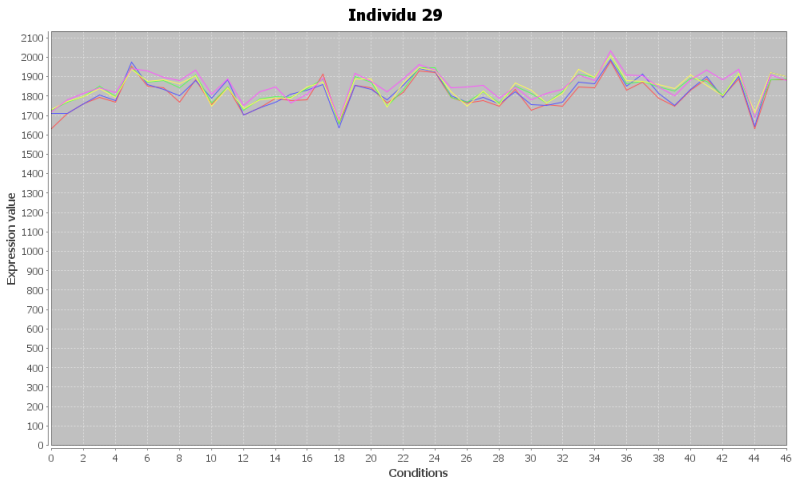

Individu 16
Expression value
Conditions

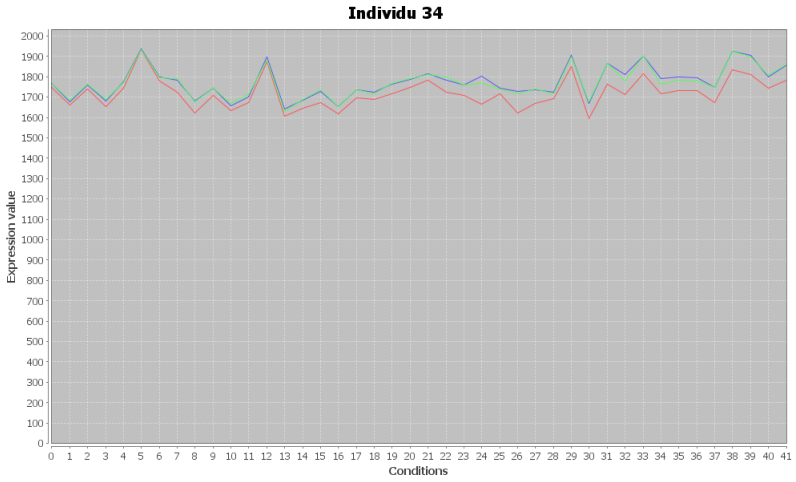

Individu 19
Expression value
Conditions

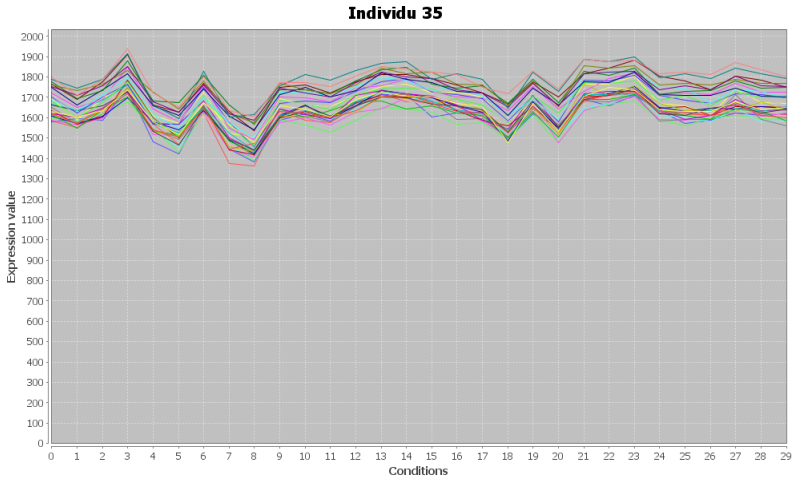

Individu 20
Expression value
Conditions

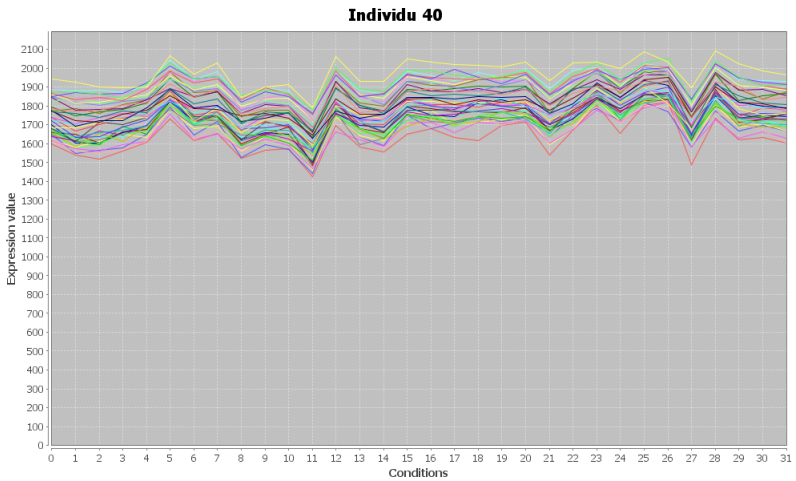

Individu 22
Expression value
Conditions

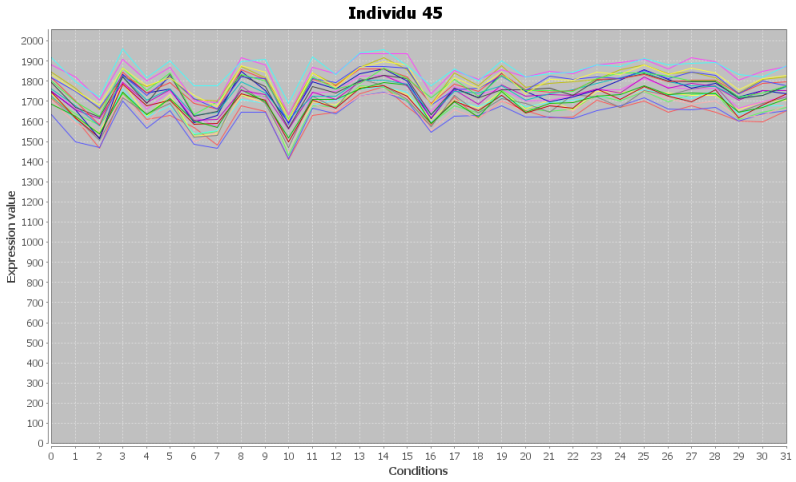

Individu 24
Expression value
Conditions

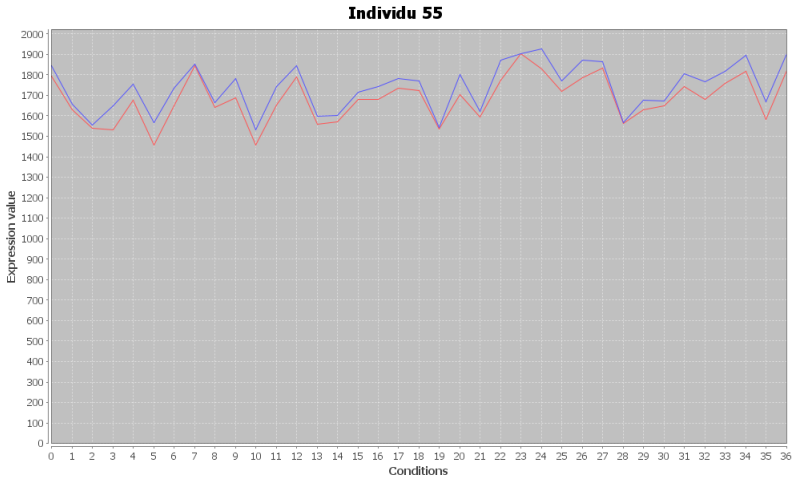

Individu 29
Expression value
Conditions

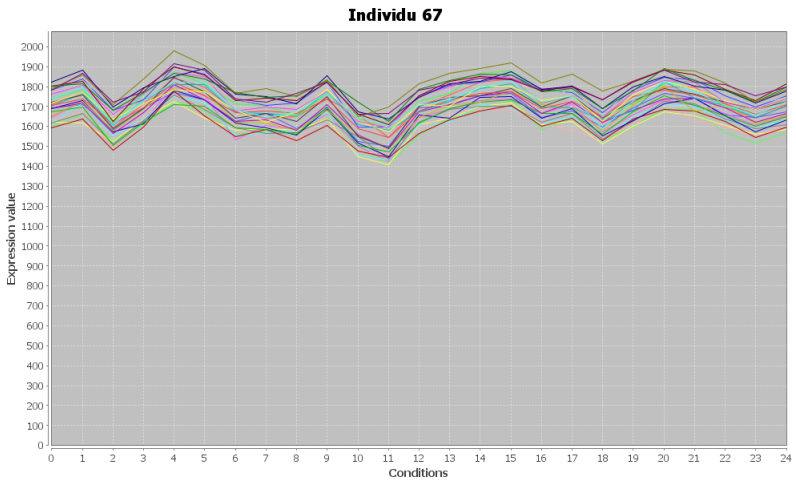

Individu 34
Expression value
Conditions

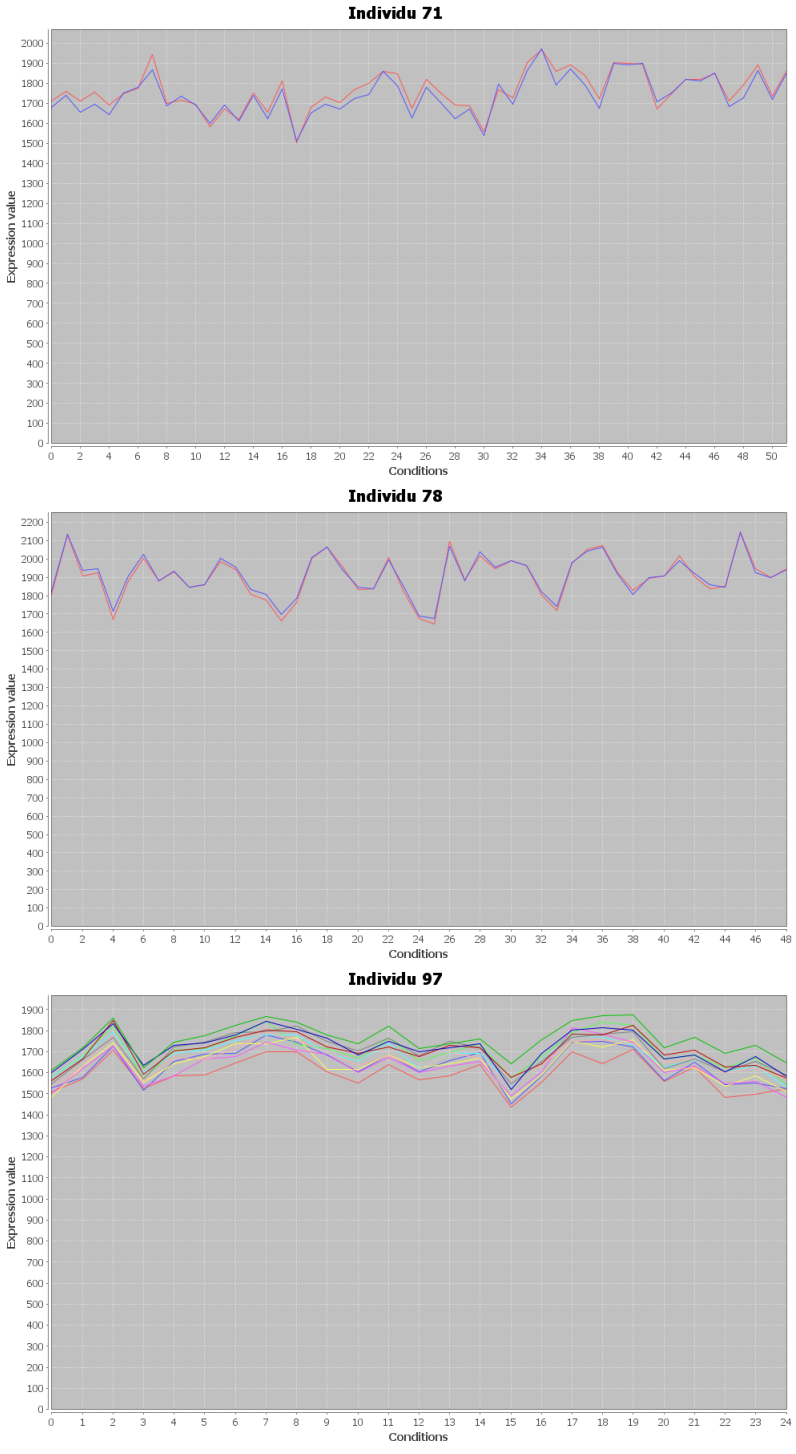

Individu 35
Expression value
Conditions


Individu 40
Expression value
Conditions


Individu 45
Expression value
Conditions


Individu 55
Expression value
Conditions


Individu 67
Expression value
Conditions


Individu 71
Expression value
Conditions
Individu 78
Expression value
Conditions
Individu 97
Expression value
Conditions

Bibliographie

[1] Madeira, S., Oliveira, A.L. : Biclustering Algorithms for Biological Data Analysis : A Survey. IEEE-ACM Trans. Comput. Biol. Bioinform. 1, pp. 24-45 (2004)

[2] Cheng, Y., Church, G.M. : Biclustering of Expression Data. In : Proceedings of the 8th Inter- national Conf. on Intelligent Systems for Molecular Biology, pp. 93-103 (2000)

[3] Mitra, S., Banka, H. : Multi-objective evolutionary biclustering of gene expression data. Pattern Recognit. 39, pp. 2464-2477 (2006)

[4] Divina, F., Aguilar-Ruiz, J.S. : Biclustering of Expression Data with Evolutionary Computation. IEEE Trans. Knowl. Data Eng. 18(5), pp. 590-602 (2006)

[5] Divina, F., Aguilar-Ruiz, J.S. : A Multi-Objective Approach to Discover Biclusters in Microarray Data. GECCO '07 : Proceedings of the 9th annual conference on Genetic and evolutionary computation. pp. 385-392 (2007)

[6] Bleuler, S., Prelic, A., Zitzler, E. : An EA framework for biclustering of gene expression data. In : Proceeding of Congress on Evolutionary Computation, pp. 166-173 (2004)

[7] Gallo, C., Carballido, J.A., Ponzoni, I. : Microarray Biclustering : A Novel Memetic Approach Based on the PISA Platform. LNCS, vol. 5483, pp. 44-55. Springer, Heidelberg (2009)

[8] Gallo, C., Carballido, J.A., Ponzoni, I. : BIHEA : A Hybrid Evolutionary Approach for Microarray Biclustering. LNCS, vol. 5676, pp. 36-47. Springer, Heidelberg (2009)

[9] fr.wikipedia.org : Génomique

[10] www.savoirs.essonne.fr : Les puces à ADN : un grand saut pour la recherche : les puces et leur utilité

[11] fr.wikipedia.org : Puce à ADN / en.wikipedia.org : DNA microarray

[12] Ren Peeters : The maximum edge biclique problem is NP-complete. Discrete Applied Mathematics, 131(3) : pp. 651-654 (2003)

[13] Johnson, D.S : The NP-completeness column : an ongoing guide. J. Algorithms 8 : pp. 438-448 (1987)

[14] Orlin, J. : Containment in graph theory : covering graphs with cliques, Nederl. Akad. Wetensch. Indag. Math. 39 : pp. 211-218 (1977)

[15] fr.wikipedia.org : Clique (théorie des graphes)

[16] fr.wikipedia.org : Partitionnement de données

[17] J.T. Tou, R.C. Gonzalez : Pattern Recognition Principles, Addison-Wesley, London (1974)

[18] E. Segal, A. Battle, and D. Koller : Decomposing gene expresision into cellular processes. In Pacific Symposium on Biocomputing, pp. 8 :89-100 (2003)

[19] Mirkin, B. : Mathematical Classification and Clustering. Dordrecht : Kluwer (1996)

[20] Hartigan, J.A. : Direct clustering of a data matrix. JASA 67 : pp. 123-129 (1972)

[21] G. Getz, E. Levine, and E. Domany : Coupled two-way clustering analysis of gene microarray data. In Proceedings of the Natural Academy of Sciences USA, pp. 12079-12084 (2000)

[22] Chun Tang, Li Zhang, Idon Zhang, and Murali Ramanathan : Interrelated two-way clustering : an unsupervised approach for gene expression data analysis. In Proceedings of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering, pp. 41-48 (2001)

[23] Stanislav Busygin, Gerrit Jacobsen, and Ewald Kramer : Double conjugated clustering applied o leukemia microarray data. In Proceedings of the 2nd SIAM International Conference on Data Mining, Workshop on Clustering High Dimensional Data (2002)

[24] Zimmermann, P., Wille, A., Buhlmann, P., Gruissem, W., Hennig, L., Thiele, L., Zitzler, E., Prelic, A., Bleuler, S. : A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics 22(9),pp. 1122-1129 (2006)

[25] Amos Tanay, Roded Sharan, and Ron Shamir : Discovering statistically significant biclusters in gene expression data. In Bioinformatics, vol. 18 (Suppl. 1), pp. S136-S144 (2002)

[26] Haixun Wang, Wei Wang, Jiong Yang, and Philip S. Yu : Clustering by pattern similarity in large data sets. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 394-405 (2002)

[27] Jinze Liu and Wei Wang. Op-cluster : Clustering by tendency in high dimensional space. In Proceedings of the 3rd IEEE International Conference on Data Mining, pp. 187-194 (2003)

[28] Laura Lazzeroni and Art Owen : Plaid models for gene expression data. Technical report, Stanford University (2000)

[29] Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller : Rich probabilistic models for gene expression. In Bioinformatics, vol. 17 (Suppl. 1), pp. S243-S252 (2001)

[30] Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller : Decomposing gene expression into cellular processes. In Proceedings of the Pacific Symposium on Biocomputing, vol. 8, pp. 89-100 (2003)

[31] Qizheng Sheng, Yves Moreau, and Bart De Moor : Biclustering micrarray data by gibbs sampling. In Bioinformatics, vol. 19 (Suppl. 2), pp. ii196-ii205 (2003)

[32] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu : δ-clusters : Capturing subspace correlation in a large data set. In Proceedings of the 18th IEEE International Conference on Data Engineering, pp. 517-528 (2002)

[33] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu : Enhanced biclustering on expression data. In Proceedings of the 3rd IEEE Conference on Bioinformatics and Bioengineering, pp. 321-327 (2003)

[34] Andrea Califano, Gustavo Stolovitzky, and Yunai Tu : Analysis of gene expression microarays for phenotype classification. In Proceedings of the International Conference on Computacional Molecular Biology, pp. 75-85 (2000)

[35] Yuval Klugar, Ronen Basri, Joseph T. Chang, and Mark Gerstein : Spectral biclustering of microarray data : coclustering genes and conditions. In Genome Research, vol. 13, pp. 703-716 (2003)

[36] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini : Discovering local structure in gene expression data : The order-preserving submatrix problem. In Proceedings of the 6th International Conference on Computacional Biology (RECOMB'02), pp. 49-57 (2002)

[37] T. M. Murali and Simon Kasif : Extracting conserved gene expression motifs from gene expression data. In Proceedings of the Pacific Symposium on Biocomputing, vol. 8, pp. 77-88 (2003)

[38] Farida Zehraoui : Fouille de données et apprentissage. Université d'Evry-Val d'Essonne, Laboratoire d'Informatique, Biologie Intégrative et Systèmes Complexes (IBISC).

[39] fr.wikipedia.org : Algorithme génétique

[40] F. Divina : Hybrid genetic relational search for Inductive Learning, PhD thesis. Department of Computer Science, Vrije Universiteit, Amsterdam, the Netherlands (2004)

[41] JP. Leclercq : Cours d'Optimisation Combinatoire et Discrète. Faculté d'Informatique, FUNDP

[42] Schaffer JD : Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In : Genetic Algorithms and their Applications : Proceedings of the First International Conference on Genetic Algorithms, pp. 93-100 Lawrence Erlbaum, Hillsdale, New Jersey (1985)

[43] Horn J, Nafpliotis N, Goldberg DE : A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In : Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol.1, pp. 82-87 IEEE Service Center, Piscataway, New Jersey (1994)

[44] Zitzler E, Laumanns M, Thiele L : SPEA2 : Improving the Strength Pareto Evolutionary Algorithm. In : Giannakoglou K, Tsahalis D, Periaux J, Papailou P, Fogarty T (eds.) EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, pp. 95-100 Athens, Greece (2001)

[45] Deb K, Pratap A, Agarwal S, Meyarivan T : A Fast and Elitist Multiobjective Genetic Algorithm : NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), pp. 182-197 (2002)

[46] Zitzler, E., Künzli, S. : Indicator-Based Selection in Multiobjective Search. In : Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiòo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832-842. Springer, Heidelberg (2004)

[47] D. E. Goldberg : Genetic Algorithms in Search,Optimization and Machine Learning. Addison Wesley, (1989)

[48] D. E. Goldberg and L. Robert : "Alleles, loci and the travelling salesman problem" in Proceedings of 1st Int. Conf. on Genetic Algorithms, J. e. Grefenstette, Ed. Lawrence Erlbaum Associates, Hillsdale, pp. 154-159 (1985)

[49] P. J. Bentley and D. W. Corne : Creative evolutionary systems. Morgan Kaufmann Publishers Inc. (2001)

[50] T. Yamada and R. Nakano : "A genetic algorithm applicable to large-scale job-shop problems" in Parallel Problem Solving from Nature, 2, R. Manner and B. Manderick, Eds. Amsterdam : Elsevier Science Publishers, B. V. (1992)

[51] D. K. Gehlhaar, G. M. Verkhivker, P. A. Rejto, C. J. Sherman, D. B. Fogel, L. J. Fogel, , and S. T. Freer : "Molecular recognition of the inhibitor ag-1343 by hiv-1 protease : conformationally flexible docking by evolutionary programming" Chemistry and Biology, vol. 2, no. 5, pp. 317-324 (1995)

[52] G. F. Spencer : "Automatic generation of programs for crawling and walking" in Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93, S. Forrest, Ed. University of Illinois at Urbana-Champaign : Morgan Kaufmann, pp. 654 (1993)

[53] F. Divina and E. Marchiori : "Evolutionary concept learning" in GECCO 2002 : Proceedings of the Genetic and Evolutionary Computation Conference. New York : Morgan Kaufmann Publishers, pp. 343-350 (1993)

[54] en.wikipedia.org : Tournament selection

[55] B. L. Miller and D. E. Golberg : Genetic algorithms, tournament selection, and the effects of noise. Complex Systems, vol. 9(3) :pp. 193-212 (1995)

[56] fr.wikipedia.org : Algorithme évolutionniste

[57] KIMURA SHUHEI, KONAGAYA AKIHIKO : "A Genetic Algorithm with Distance Independent Diversity Control for High Dimensional Function Optimization." in Transactions of the Japanese Society for Artificial Intelligence, vol. 18, pp. 193-202 (2003)

[58] Antonio Lopez Jaimes and Carlos A. Coello Coello : "An Introduction to Multi-Objective Evolutionary Algorithms and Some of Their Potential Uses in Biology" in Applications of Computational Intelligence in Biology, vol. 122 pp. 79-102 (2008)

[59] Calonder M., Zitzler E. : "Multi-objective clustering of gene expression data with evolutionary algorithms." Master Thesis at Systems Optimization Group, ETH Zurich. ETH Library (2006)

[60] Edgeworth FY. : Mathematical Psychics. P. Keagan, London, England (1881)

[61] Pareto V. : Cours d'Economie Politique, vol. 1 et 2 F. Rouge, Lausanne (1896)

[62] Osyczka A. : Multicriteria optimization for engineering design. In : Gero JS (ed.) Design Optimization,pp. 193-227. Academic Press (1985)

[63] Carlos A. Coello Coello : "An Updated Survey of Evolutionary Multiobjective Optimization Techniques : State of the Art and Future Trends" in Proceedings of the Congress on Evolutionary Computation, IEEE Press, pp. 3-13 (1999)

[64] Coello Coello CA, Lamont GB (eds.) : "Applications of Multi-Objective Evolutionary Algorithms." World Scientific, Singapore. (2004)

[65] Blum C, Roli A : Metaheuristics in combinatorial optimization : Overview and conceptual comparison. ACM Computing Surveys, vol. 35(3), pp. 268-308 (2003)

[66] Deb K. : "Multi-Objective Optimization using Evolutionary Algorithms." John Wiley & Sons, Chichester, UK. (2001)

[67] Ehrgott M. : "Multicriteria Optimization." Springer, Berlin, second edn. (2005)

[68] R. Cho, M. Campbell, E. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. Wolfsberg, A. Gabrielian, D. Landsman, D. Lockhart, and R. Davis. : A genome-wide transcriptional analysis of the mitotic cell cycle. Molecular Cell, 2 : pp. 65-73 (1998)

[69] A. A. Alizadeh and et al. : Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. Nature, 403 : pp. 503-511 (2000)

[70] Tavazoie, S., Hughes, J.D., Campbell, M.J., Cho, R.J., and Church, G.M. : "Systematic determination of genetic network architecture." Nature Genetics 22 : pp. 281-285 (1999)

[71] Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., Levine, A. : "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays." Proc. Natl. Acad. Sci. 96, pp. 6745-6750 (1999)

[72] fr.wikipedia.org : Cycle cellulaire

[73] fr.wikipedia.org : Saccharomyces cerevisiae

[74] http ://arep.med.harvard.edu/biclustering/

[75] http ://microarray.princeton.edu/oncology/affydata/index.html

[76] fr.wikipedia.org : Portail de la Biologie cellulaire et moléculaire

[77] http ://vortex.cs.wayne.edu/

[78] http ://search.cpan.org/ sherlock/